Förbättrad karaktärtrovärdighet i dataspel med hjälp av målorienterat beteende

Alex Blomqvist

Innehåll:

1.Introduktion

Artificiell intelligens i dataspel handlar om att förbättra spelarnas upplevelser. Det är särskilt viktigt eftersom utvecklaren levererar spelupplevelser till en växande publik. Dataspel har längs med tiden blivit mer tillgängliga genom ett brett utbud av mobila apparater. Teknologin samt AI har möjliggjort för utvecklare att leverera konsolliknande upplevelser till nya plattformar. Därför skulle element för goda narrativ, så som karaktärtrovärdighet, kunna förbättra spelupplevelsen då fler spelare vill fördjupa sig i spelen. Karaktärtrovärdighet är väsentlig för spel som strävar efter en imponerande immersion och för att få spelaren investerad i spelets narrativa aspekter så som budskap och tema. Alla spel behöver inte fokusera sig narrativ men spelvärlden oftast bebor karaktärer och på så sätt måste ta i beaktan karaktärtrovärdighet. I framtiden ser jag att AI kommer användas alltmer för narrativt och emotionellt engagemang i spelen.

Syftet med min avhandling är att göra en redogörelse för artificiell intelligens och planbaserat målorienterat beteende för karaktärer inom dataspel, varefter denna teknik jämförs med andra lösningar som används av spelindustrin. Analysens huvudfokus är dataspelskaraktärernas karaktärtrovärdighet. Genom att läsa arbetet borde läsaren förstå nyttorna och problemen med målorienterad spel-AI och få en baskunskap om karaktärtrovärdighet inom AI-lösningar.

Avsnitt 2 presenterar idén bakom karaktärerna ur narratologins perspektiv varefter konceptet behandlas ur AI-perspektiv. En stegvis representation av den centrala teorin bakom artificiell intelligens ges. De olika delområdena beskrivs och för varje kategorisering ges ett exempel på dataspel där den används. Nästa avsnitt handlar om planerbaserat målorienterat beteende och de mest centrala metoderna för att kunna implementera det. Slutligen i avsitt 4 jämförs målorienterad artificiell intelligens med andra sätt så som beteende träd och nyttoinriktad AI. Dessutom listas olika för- och nackdelar som målorienterad artificiell intelligens har.

2. Karaktär och artificiell intelligens

2.1 Narratologiska begrepp

Återgivande av en serie händelser kallas för en berättelse eller ett narrativ. En berättelse handlar om någonting och består av handlingar som är relevanta för detta. Berättelsen beskriver hur ett visst tillstånd förändras till ett annat och består av ett händelseförlopp tills berättelsen är slut. En berättelse kan behandla verkliga eller fiktiva händelser och en blandning av båda. [Prince]

En aktör är något, oftast en person, som har förmågan att påverka tillståndet i en berättelses värld. Man talar oftast om karaktärer istället för aktörer. Skillnaden är att ”karaktär” specificerar aktören som en person eller någon med mänskliga drag som sysslar med mänskliga uppgifter. Karaktärer består av attribut och funktioner. Ett attribut är en egenskap hos karaktären medan en funktion är en utförbar operation. [Prince, Riedl]

Karaktärstrovärdighet betyder att handlingarna som utförs av karaktären motiveras av dess övertygelser, önskningar och egenskaper [Riedl]. Detta innebär att karaktärens förmågor och liv inte behöver vara verklighetstrogna för att vara karaktärstrovärdiga. Därför kan en karaktär som kastar eldbollar med hjälp av magi vara mer karaktärstrovärdig i en fantasivärld än en karaktär som inte använder magi. Trovärdigheten beror alltså på karaktärens beteende som borde reflektera dess givna personlighet, bakgrund och erfarenheter i världen de befinner sig i. [Klug]

Då en spelare interagerar med en karaktär skapas det en bild om karaktären i spelarens huvud. Detta innebär att spelaren kommer i fortsättningen att ha vissa förväntningar om karaktärens beteende. En av dessa förväntningar är intelligens även om karaktärens inte har specifika egenskaper gällande dess sätt att tänka. Det betyder att av mänskliga varelser förväntas ha en mänsko-lik nivå av intelligens. Om karaktären är trovärdig borde spelaren kunna resonera dess avsikter och motiv genom att observera den. När en karaktär inte reagerar som spelaren hade tänkt sig eller tar ett beslut som inte passar dess personlighet förlorar karaktären sin trovärdighet. [Klug, Riedl]

2.2 Varför målorienterade karaktärer?

Det är mycket subjektivt vad en publik tycker om en berättelse som presenteras för dem. Vissa egenskaper i en god berättelse kan ändå vara universella. För att en berättelse kunde fånga publikens intresse borde berättelsen handla om ämnen som publiken kunde bli intresserade av. Berättelser borde alltså ha kvaliteter som gör dem värda att berättas. berättelser brukar bestå av problem, konflikt och upplösning. Dessa element skapar dramatik som oftast finns i goda berättelser. Det är också mycket viktigt att karaktärerna som upplever denna drama är intressanta och trovärdiga.

En karaktärs önskemål är en viktig del i karaktärsskapandet, speciellt då man önskar bra antagonister. I dataspel är det mycket vanligt att man möter antagonister på grund av dataspelens struktur och att spelets narrativ mycket sällan är konfliktfritt. Ändå är det lika viktigt att de andra karaktärerna i spelet också har övertygelser och önskningar.

De mest kända karaktärerna brukar vara extremt motiverade att uppnå sina mål. Då man har en motiverad karaktär vill man oftast att karaktären skall också vara trovärdiga. Då måste karaktären ha en uppfattning om hur den kan uppnå dessa mål och på så sätt klara av problemlösning. På så sätt bevisar karaktären sin motivation för publiken, som i dessa fall är spelaren. Enligt Riedl och Young [Riedl] anses en berättelse mer trovärdig om berättelsevärldens karaktärer verkar vara motiverade av individuella mål och önskningar.

2.3 Agenter

Ett objekt som uppfattar sin omgivning med hjälp av sensorer och inverkar på denna med aktuatorer kallas för en agent. Aktuator eller ställdon är alltså verktyg för att utföra uppgifter som en mekanism har. Då ett system har en eller fler sensorer borde det också ha en eller fler aktuatorer för att sensorerna själva får inget att hända. Uppfattningen agenten har vid en viss tidpunkt heter percept. En perceptserie hos en agent är historiken över allt som agenten uppfattat. En agents beteende beskrivs av en agentfunktion (en abstrakt matematisk beskrivning) som avbildar en given perseptserie till en handling. Internt kommer agentfunktionen för en artificiell agent att implementeras av ett agentprogram (se avsnitt 2.5). [Russell]

I dataspel är det väldigt vanligt att man möter som använder en eller fler av industriens standardsensorer. Standardsensorerna inkluderar radar, objektintervallfyndare, eldlinje- och målsensor. Radarn används ofta för att ge agenten en uppfattning var en vän eller en fiende befinner sig, medan objektintervallfyndare används för att räkna distansen till olika objekt. Målsensorer används oftast i skjutspel för att en agent skall väta när den kan skjuta möjliga fienden, medan eldlinjesensorer används i samma spel för att informera agenten om möjliga vänner som ligger på eldlinjen så att den kan undvika vådabekämpning. I alla fall aktuatorerna kan variera betydligt, men de allra vanligaste aktuatorer man möter på är agentens rörelseknappar.

Inom artificiell intelligens motsvarar en agent narratologins aktör. I likhet med en aktör är en agent ett objekt som har förmågan att förändra tillståndet i en berättelsevärld. En agent består också helt och hållet av attribut och funktioner. Ett attribut är en egenskap hos agenten, medan en funktion är en operation som agenten kan utföra. [Riedl]

Agentbaserad artificiell intelligens i dataspel handlar om att producera autonoma karaktärer som tar in information från speldatat, bestämma vilka åtgärder som ska exekveras baserat på informationen och genomföra dessa handlingar. [Millington]

Enligt Russell och Norvig [Russell] är en rationell agent en agent som för varje möjlig perseptserie väljer den handling som förväntas maximera dess prestandamått, med tanke på informationen från perseptserien och den inbyggda kunskap som agenten har. Ett prestandamått som definierar kriteriet för agentens framgång. Måttet tilldelar poäng för sekvenser på olika tillstånd, så att agenten skulle kunna urskilja nyttan för olika handlingsval. Agenten måste alltid agera för att uppnå det bästa resultatet eller det bästa förväntade resultatet, när det finns osäkerhet. En rationell agent borde också vara autonom, vilket betyder att agenterna ska förlita sig på sina egna percept hellre än förhandskunskap av sin skapare, det vill säga den ska lära sig så mycket som möjligt för att kompensera för eventuella brister eller felaktiga förhandskunskaper.

En agent kan utnyttja ett planbaserat system (se avsnitt 3.1) för att hitta en serie handlingar för att uppnå sina mål. Detta kan utföras genom att söka i en tillståndsrymd (se avsnitt 3.3.1) med en sökalgoritm (se avsnitt 3.3.2).

2.4 Agentomgivningar

En omgivning är allt i spelvärlden som omger agenten, men är inte en del av själva agenten. En omgivning kan beskrivas som en situation där agenten är närvarande. Omgivningens egenskaper är ett av de absolut viktiga elementen för att bestämma rätt modeller för en AI-lösning. Att förstå egenskaperna hos uppgiftsomgivningen är en av de första saker som personer som arbetar med artificiell intelligens fokuserade på för att hantera ett specifikt problem. Det förekommer ett stort antal uppgiftsomgivningar som kan kategoriseras enligt några dimensioner. Dessa dimensioner avgör hur man ska utforma och implementera agenten.

En uppgiftsomgivning är antingen fullständigt eller delvist observerbar. I en fullständigt observerbar uppgiftsomgivning har agenten tillgång till all information som krävs för att slutföra måluppgiften. Då sensorerna på agenten inte har tillgång till all information är omgivningen delvist observerbar. Detta gäller också då omgivningen är icke-observerbar. De flesta dataspelen är delvist observerbara. Om uppgiftomgivningen för spelaren är delvist observerbar men för de artificiella karaktärerna fullständigt observerbart kunde det tolkas som att spelet fuskar.

Deterministisk eller stokastisk är dimensioner som en uppgiftsomgivning är också uppdelad i. En deterministisk uppgiftsomgivning är en på vilken resultatet kan bestämmas baserat på ett specifikt tillstånd. Med andra ord ignorerar deterministiska uppgiftsomgivningar osäkerhet. De flesta situationer anses vara stokastiska eftersom de är så komplicerade.

En dimension kategoriserar uppgiftsomgivningar som antingen statiska eller dynamiska. I en statisk omgivning kan agenten lita på datakällor som inte ändras ofta. Detta innebär att ifall omgivningen kan förändras medan agenten överväger sin nästa handling är omgivningen dynamisk. De flesta dataspel har en dynamisk uppgiftsomgivning. Men statiska uppgiftomgivningar kommer emot i spel där omgivningen kan stå stilla då agenten försöker hitta en plan.

Man kan också uppdela uppgiftomgivningar i episodiska eller sekventiella omgivningar. I en episodisk omgivning finns det en serie handlingar varav agenten utför en enda handling, och endast perceptet krävs för att bestämma handlingen. Men i en sekventiell omgivning krävs att agenten har ett minne av tidigare handlingar, det vill säga tillgång till perceptserien för att bestämma de bästa åtgärderna. I sekventiella omgivningar påverkar tidigare handlingar på agentens beslut, det vill säga varje beslut kan påverka alla framtida beslut. Dataspelens omgivningar är nästan alltid sekventiella. Dessa två skall inte blandas med turbaserade och realtids omgivningar. Turbaserade spel går ut på att sätta agenternas handlingar i en ordning och är populära bland strategispel. Delvist turbaserade spelmekaniker är sådana där ett realtidspel kan stoppas för att göra val eller att bestämma handlingsserier och sedan fortsättas i realtid. Dataspel som har segment där spelet är turbaserat och resten av spelet är i realtid är också delvist turbaserade. Realtidsspel är dataspel där spelets gång och kommandon sker i realtid. [Gunn Russell]

En uppgiftsomgivning är antingen diskret eller kontinuerlig. då omgivningen är kontinuerlig måste agenten lita på okända och snabb föränderliga datakällor, medan diskreta omgivningar är sådana där en begränsad mängd möjligheter kan påverka det slutliga resultatet av uppgiften. Detta innebär att diskreta omgivningar förevisar ett ändligt antal olika tillstånd, handlingar och persept.

Dessutom kan uppgiftsomgivningen vara monoagent- eller multiagentbaserad. Om det finns flera än en agent i omgivningen är den multiagentbaserad. Om spelet ska inkludera artificiella karaktärer eller ska kunna spelas samtidigt med andra spelaren så är spelvärlden multiagentbaserad. Exempelvis agentomgivningen i förstapersonssjutaren Quake II (Carmack et al. 1997) av Id software är delvis observerbar, stokastisk, dynamisk, sekventiell, kontinuerlig och multiagentbaserad. Denna kombination på dimensioner gäller för de flesta dataspelen.

2.5 Agentprogram

Ett agentprogram är en konkret implementering som körs inom ett fysiskt system. En agent består av ett agentprogram och en arkitektur (se avsnitt 2.6). I praktiken gör arkitekturen percepten från sensorerna tillgängliga för programmet, kör programmet och matar programmets handlingsval till aktuatorer när de genereras. Agentprogrammet som ansvarar för att implementera agentfunktionen måste vara lämpligt för den valda arkitekturen.

Skillnaden mellan agentprogrammet och agentfunktionen är att agentprogrammet använder endast det nuvarande perseptet medan agentfunktionen använder hela perseptserien som indata. Agentprogrammet arbetar endast med de nuvarande perseptet som input eftersom ingenting annat finns tillgängligt i omgivningen. För att en agent ska kunna utföra handlingar som kräver en perseptserie måste agenten komma ihåg alla persept. [Russell]

2.5.1 Enkla reflexagenter

Enkla reflexagenter agerar endast utifrån det nuvarande perseptet och ignorerar resten av sin perseptserie. Agentfunktionen är baserad på villkorshandlingsregeln: ”om villkor så handling”. Denna typ av agent är den enklaste så deras intelligens är mycket begränsad.

Den optimala omgivningen för enkla reflexagenter är fullt observerbar. Vissa reflexagenter kan också innehålla information om sitt nuvarande tillstånd som gör att de kan bortse förhållanden där en handling redan aktiverats. Oändliga slingor är ofta oundvikliga för enkla reflexagenter som arbetar i delvist observerbara omgivningar. Exempelvis i ett rutnät kan agenten få kommandot ”gå höger” i rutan A och kommandot ”gå vänster” i rutan B som ligger bredvid (höger om) rutan A. Endast om agenten slumpmässigt kan utföra sina handlingar kan det vara möjligt för agenten att komma undan från oändliga slingor. [Russell]

Inom dataspel förekommer enkla reflexagenter bland annat i äldre rollspel (RPG), som till exempel Pokémon Red (Masuda 1996/1999) av Gamefreak. I detta spel vandrar statistkaraktärerna slumpmässigt runt eller stå stilla i ett visst område i spelvärlden. När man interagerar med dem så svarar de med en av fler möjliga dialoger. I strid väljer en enkel motståndare slumpmässigt en attack medan avancerade motståndare väljer attacken enligt en viss villkorshandlingsregel. Enkla reflexagenter är användbara i situationer där en hög nivå av komplexitet inte krävs av en deltagare. Ju mer begränsat omfattningen av möjliga handlingar är till exempel i en diskret actionspelvärld, desto mindre komplexitet krävs vid beslutsfattande. [Gunn]

2.5.2 Modellbaserade reflexagenter

En modellbaserad reflexagent är det effektivaste sättet att hantera delvist observerbara omgivningar. Omgivningens nuvarande tillstånd lagras i agenten som upprätthåller någon form av struktur som beskriver den del av världen som inte kan ses. Det betyder att agenten har en information om hur världen utvecklas utan agenten och hur agentens egna handlingar påverkar världen. Denna kunskap om ”hur världen fungerar” kallas en världsmodell, därav namnet modellbaserad agent.

En modellbaserad reflexagent bör upprätthålla ett internt tillstånd som beror på perceptserien och därigenom återspegla åtminstone några av de obemärkta aspekterna av det nuvarande tillståndet. Perseptserien och effekterna av åtgärder på miljön kan bestämmas med hjälp av detta interna tillstånd. Världsmodellen möjliggör att agenten kan uppdatera interna tillståndet. Agenten väljer sedan en handling på samma sätt som en enkel reflexagent. [Russell]

En agent kan också använda modeller för att beskriva och förutse beteenden hos andra agenter i omgivningen. Denna typ av agent skulle bäst tillämpas i dynamiska realtidsspel där konstant övervakning av miljön krävs för att basera beslut om handlingar. Modellbaserade reflexagenter är också ett bra val för samarbetsspel. [Gunn]

2.5.3 Målinriktade agenter

Målinriktade agenter utvidgar intelligensen som hittas i modellbaserade reflexagenter genom att använda information om sina mål. Målinformationen beskriver önskvärda situationer. Detta gör det möjligt för agenten att välja mellan flera möjligheter. Agenten kommer att välja den som når ett måltillstånd. Ibland är agenten tvungen att överväga långa handlingsserier för att uppnå dessa mål. För att undvika långa handlingsserier som är tidskrävande att hitta kan systemet begränsa handlingsseriernas längd. Sökning (se avsnitt 3.3) och planering (se avsnitt 3.1) är delområden i artificiell intelligens som ägnas åt att hitta handlingsserier som uppnår agentens mål. Detta gör agenten kapabel för problemlösning. [Russell]

I dataspel används målinriktade agenter som till exempel i de moderna Tomb Raider-spelen som utvecklats av Crystal Dynamics. Denna nivå av oförutsägbart beteende är mycket komplicerad i en tillståndsmaskin (se avsnitt 3.4), men ett planeringssystem innebär att en NPC gör det automatiskt. Målinriktade agenter är speciellt fördelaktiga i otillgängliga spelvärldar, eftersom de kan ändra sina egna delmål då informationen de har förändras. [Gunn]

2.5.4 Nyttoinriktade agenter

Målinriktade agenter skiljer endast mellan måltillstånd och icke-måltillstånd. Det är möjligt att definiera ett mått utifrån hur önskvärt ett visst tillstånd är. Detta prestandamått kan erhållas genom att använda en nyttofunktion som beräknar olika tillstånds nytta så att agenten kan jämföra de olika handlingarna och välja den med tanke på nyttan. Ett mer generellt resultatmått bör möjliggöra jämförelse av olika världstillstånd beroende på exakt hur nöjd de skulle göra agenten. Termen nytta kan användas för att beskriva hur nöjd agenten är.

En rationell nyttoinriktad agent väljer den handling som maximerar den förväntade nyttan av resultaten. Agenten utgår alltså ifrån vad den i genomsnitt förväntar sig att få ut med tanke på sannolikheten och nyttan för varje utfall. En nyttoinriktad agent måste modellera och hålla reda på sin omgivning för att kunna fungera på rätt sätt. Med en explicit nyttofunktion kan agenten fatta rationella beslut med hjälp av någon generell algoritm som inte är beroende av att nyttofunktionen maximeras. [Russell]

Nyttoinriktade agenter är skärkilt användbara i managerspel eller i delvist observerbara realtidsspelvärldar. I dessa fall kan man tillämpa det ständigt förändrande tillståndet i den observerbara spelvärlden på de olika målen när informationen blir tillgängligt och sedan ändra beteende gradvis. [Gunn]

Ett exempel på nyttoinriktade agenter i dataspel är karaktärerna i The Sims (Wright et al. 2000) av Maxis. Karaktärerna i spelet kan välja handlingar som förbättrar deras humör automatiskt utan spelarens kommando. Humöret beräknas genom att addera åtta olika behov i en formel. Dessa behov är ”hunger”, ”energi”, ”komfort”, ”kul”, ”hygien”, ”umgänge”, ”blåsa” och ”rum”. Formeln gör att

behov som är större prioriteras. Det borde i sin tur ge karaktären automatiska handlingar som är kritiska, men i praktiken kan det vara så att omgivningen saknar objekt som tillåter dessa handlingar. I The Sims finns det också scenarion där nyttan inte beaktas, exempelvis betalar simsen aldrig sina räkningar automatiskt även om det leder till större problem för dem. Detta sker för att spelet inte anser betalade räkningar som ett behov, viket i sin tur ger spelaren kontroll över simsen. Karaktärens egenskaper kan lägga mer vikt på vissa behov, vilket orsakar att olika karaktärer kan ha en tendens för att börja olika handlingar då de är automatiskt valda.

2.5.5 Agenter med inlärningsförmåga

De tidigare nämnda agenterna refereras nuförtiden till gammaldags AI eftersom nuvarande studier inom AI handlar, för det mesta, om agenter med inlärningsförmåga. Det som gör agenter med inlärningsförmåga unika är att alla de tidigare nämnda agenterna kan vara sådana. En agent med inlärningsförmåga består av fyra element: inlärningselement, prestandaelement, kritikerelement och problemgenerator.

Inlärning har fördelen att den tillåter agenterna att initialt arbeta i okända omgivningar och att bli mer kompetenta än deras ursprungliga kunskap skulle tillåta. Största skillnaden gentemot andra agenter finns i växelverkan med inlärningselementet och prestandaelementet. Inlärningselementet ansvarar för förbättringar medan prestandaelementet ansvarar för att välja externa handlingar.

Inlärningselementet använder information från kritikerelementet om hur agenten utför sin uppgift och bestämmer hur prestandaelementet ska modifieras för att utföra uppgiften bättre i framtiden. Prestandaelementet är vad andra agenterna har beaktat som hela agenten. Kritikerelementet är nödvändigt eftersom perceptet inte ger en indikation om agentens framgång. Till exempel, en schackspelande NPC kan få en uppfattning om att den har försatt sin motståndare i schackmat, men den behöver en prestandastandard för att veta att detta är bra eftersom själva uppfattningen säger inte det. Dessutom är det viktigt att prestandastandarden är fastställd. Oftast tänker man att prestandastandarden inte tillhör agenten eftersom agenten själv får aldrig modifiera prestandaelementet enligt dess beteende. [Russell]

Den sista komponenten i inlärningselementet är problemgeneratorn. Den är ansvarig för att föreslå handlingar som kommer leda till nya och informativa upplevelser. Idén går ut problemgeneratorn testar de olika suboptimala handlingsalternativen (enligt den information som är tillgänglig) för att hitta möjliga handlingar som i framtiden eller längs med tiden är bättre. [Russell]

Agenter med maskininlärning används mycket lite i kommersiella dataspel. En orsak kan vara att dataspel har varit så framgångsrika att en ny teknik som maskininlärning, som skulle fundamentalt förändra spelupplevelsen, kan uppfattas som en riskfylld investering av branschen. Budgetarna för de så kallade AAA- spelen är såpass stora att de flesta spelbolagen endast arbetar med konservativa tillvägagångssätt. AAA är en klassificering för dataspel som publicerats av stora och medelstora datorspelsförlag.

2.6 Agentarkitektur

2.6.1 BDI

2.6.2 Andra alternativ

3. Målorienterat beteende

3.1 Planering

Med AI-planering menar man en explicit överläggningsprocess som väljer och organiserar åtgärder genom att förutse deras resultat. Agenten som använder planering strävar efter att uppnå vissa förutbestämda mål. I praktiken går planering ut på att ett system försöker hitta olika serier av handlingar för att uppnå ett visst avlägset mål som systemets skapare valt. Den valda handlingsserien kallas för en plan.

För att kunna skapa en planerar måste man modellera problemet i det språk som systemet använder. Modelleringen ska möjliggöra systemets tillgång till designerns skapade predikat som beskriver spelvärlden. Dessa predikat är alltså fakter gällande spelvärlden som agenten kan använda för att göra beslut. De olika kombinationerna av dessa predikat kallas för tillstånd medan de gällande predikat kallas för världstillståndet. Ett predikat kan vara till exempel ammunition. Då ”ammunition” stämmer kan karaktären använda sitt vapen men när ”ammunition” är osant måste karaktären skaffa ammunition för att kunna använda sitt vapen. Om världen har flera vapen och tilläggs ammunition är bunden till karaktären måste predikatet specificera ”karaktär A har ammunition” och ”vapen B har ammunition”. Då om karaktär A har vapnet B och predikatet ”vapen B har ammunition” är osant medan ”karaktär A har ammunition” är sant, måste A ladda B med sin ammunition för att använda B. Sedan om man vill att viss ammunition fungerar med endast specifika vapen måste predikatet specificera vilken typ av ammunition. Anors i ett fall där vapen B är en pistol och vapen C en kanon skulle ammunition fungera lika bra med B och C.

För att bygga upp handlingar som spelets artificiella agenter skulle kunna utföra måste handlingen innehålla information om tre grundläggande fakta. Första information som borde vara tillgängligt är objekt som involveras i handlingen. Förhandsvillkoren för handlingen är den nästa informationsbiten som ska vara tillgängligt. De tredje informationen som ska vara tillgänglig är effekten från handlingen. Detta innebär hur predikaten i världstillståndet förändras när handlingen utförs.

Ett program som väljer ut en plan för agenten heter planerare. [Russell]

3.2 STRIPS

Stanford Research Institute Problem Solver eller STRIPS är en planerare som utvecklats av Richard Fikes och Nils Nilsson år 1971. Planeraren användes av Shakey the robot, den första rörliga roboten med synförmåga. Projektet var det första som sammanfogade logiskt resonemang med fysisk handling. STRIPS-språket antar att världen är sluten. Detta innebär att alla uttalanden som inte nämns i ett tillstånd antas vara falska.

3.3 Sökning

En agent observerar sin uppgiftsomgivning genom sina sensorer och agentprogrammet representerar den här observationen som ett tillstånd. Då man skapar en algoritm för att uppnå ett givet måltillstånd kan agenten söka igenom sekvenser av liknande tillstånd ända tills man hittar en plan. För att en sökning ska vara möjligt måste man veta initialtillståndet, alla möjliga handlingar, måltillståndet och vägkostnaderna.

Sökning antar att uppgiftsomgivningen är diskret, deterministisk, statisk och fullt observerbar [Länk]. Men de flesta uppgiftsomgivningarna i dataspel är kontinuerliga, stokastiska, dynamiska och delvist observerbara. Detta problem löses med hjälp av strategi och dynamisk sökning. Då ett system använder dynamisk sökning kontrolleras världstillståndet när en plan hittas. För att sökningen är giltig borde världstillståndet vara samma som initialtillståndet. Om sökningen är ogiltig måste agenten utföra en ny sökning. När sökningen är giltig utför karaktären den valda planen så länge som ingen handling i handlingsserien annulleras. En handling eller handlingsserie kan annulleras på grund av ett upptäckt hinder eller om ett nytt mål prioriteras högre än målet som den nuvarande planen försöker uppnå. Då måste agenten söka en ny plan, om målet är samma som tidigare kommer agentens information om möjliga hindret hjälpa sökningen. Planeraren kommer att föreslå en ny handlingsserie som inte inkluderar handlingen som haft ett hinder.

Det finns olika strategier för sökning. Detta innebär att man kan redigera sökningen enligt behoven som systemet har. I allmänhet finns det två grundprinciper: bredden först- sökning och djupet först-sökning. Men då sökrymden är stor, använder man sig av heuristiker. Med en heuristik menar man

en metod eller algoritm för att lösa ett beräkningskomplext problem snabbare när de klassiska metoderna är långsamma, eller för att hitta en ungefärlig lösning när de klassiska metoderna misslyckats med att hitta en exakt lösning. Denna princip kallas för Bäst först- sökning och är basen för den optimala AI-sökalgoritmen (se avsnitt 3.3.2) A*.

3.3.1 Tillståndsrymden

Tillståndsrymden består av alla världstillstånd som det finns.

3.3.2 Sökalgoritm

A* är en girig bäst först- algoritm som påminner om principen för djupet först- sökning.

3.4 Finita tillståndsmaskiner

En finit tillståndsmaskin är en beräkningsmodell som baserar sig på en hypotetisk maskin som består av en eller fler tillstånd.

3.5 GOAP

Goal-Oriented Action Planning eller GOAP är ett STRIPS-baserat planeringssystem som utvecklats av Monolith Productions för förstapersonsskjutaren First Encounter Assault Recon eller F.E.A.R.. GOAP-systemet drivs av en enkel tillståndsmaskin, med tre olika tillstånd. Dessa tillstånd är ”Gå till”, ”Använd smart objekt” och ”Tomgångsanimering”, och alla tillstånd är tillgängliga från alla tillstånd. I praktiken spelar varje tillstånd en animering för att då till exempel en karaktär rör på sig så avbildar systemet en rörelseanimation för karaktären. Samma gäller för användning av smarta objekt som kan vara så enkla som öppna en dörr, och för tomgångsanimering som kan variera från att karaktären märkt något till att karaktären avfyrar sitt vapen. Hela systemet exekveras i en mycket mindre och datadriven implementation.

Systemet skapar plan som löser specifika problem genom att använda A*- algoritmen. Handlingarna i planen är översatta till specifika tillstånd i tillståndsmaskinen med data som till exempel platser att röra sig emot, animationer att spela och smarta objekt att interagera med. [Orkin]

3.6 HTN

GOAP är inte den enda avancerade lösningen på ett handlingsplaneringsproblem. Hierarchical Task Network eller HTN är ett tillvägagångssät för AI-planering där beroendet mellan handlingar kan ges i form av hierarkiskt strukturerade nätverk. Killzone 2 (van der Leeuw 2009) av Guerrilla Games populariserade HTN som ett möjligt alternativ till GOAP-baserade spel.

HTN introducerar tre olika typer av handlingar: målhandlingar, primitiva handlingar och sammansatta handlingar. Målhandlingar och primitiva handlingar påminner om de handlingar som tidigare uppkommit i STRIPS-baserade planerare. HTN är ännu mer resurstung än STRIPS-liknade tillvägagångssätt. Till exempel i Killzone 2 körs planeraren bara fem gånger per sekund. [] Dessutom är HTN-planerare mycket komplexa och de är därför bara praktiska för medelstora och stora spelutvecklingsstudior.

4. Jämför med spel-AI

4.1 Alternativ som inte är målorienterade

4.1.1 Beteendeträd

4.1.2 Andra alternativ

4.2 Karaktärtrovärdighet

4.3 Andra fördelar

4.4 Nackdelar och problem

5. Slutord

Litteraturlista:

[] J. Lebowitz, C. Klug; Interactive storytelling for video games: A player-centered approach to creating memorable characters and stories; s. 88-89; Oxford: Focal Press (Elsevier); 2011.

[] I. Millington, J. Funge; Artificial intelligence for games, 2nd Ed.; s. 11; CRC press; 2009

[] G. Prince; A Dictionary of Narratology Revised Edition; s. 1-4 & 58-60; University of Nebraska Press; 2003

[] E. A. A. Gunn, B. G. W. Craenen, E. Hart; A Taxonomy of Video Games and AI; AI and Games Symposium; s. 10,

[] M. O. Riedl; Narrative Planning: Balancing Plot and Charecter; ;2004