

STYRBETEENDEN FÖR ATT NAVIGERA GRUPPER AV AUTONOMA AGENTER

STEERING BEHAVIORS TO NAVIGATE GROUPS OF AUTONOMOUS AGENTS

Examensarbete inom huvudområdet Datavetenskap
Grundnivå 30 högskolepoäng
Vårtermin 2015

Arvid Backman

Handledare: Mikael Johannesson
Examinator: Anders Dahlbom

Sammanfattning

Nyckelord:

Innehållsförteckning

1	Introduktion.....	1
2	Bakgrund.....	2
2.1	Artificiell intelligens.....	2
2.1.1	Traditionell AI	2
2.1.2	Spel-AI.....	2
2.1.3	Autonom agent	2
2.2	Styrbeteende	3
2.2.1	Sök	3
2.2.2	Ankomst	4
2.2.3	Väggundvikande.....	5
2.2.4	Vägföljning.....	5
2.2.5	Flödesfält.....	5
2.2.6	Flockbeteende.....	6
2.2.7	Separation	6
2.2.8	Sammanhållning.....	7
2.2.9	Formering	7
2.3	Beräkningsmodell för kombination av styrbeteenden	8
2.3.1	Viktad trunkerad summa	8
2.3.2	Viktad trunkerad summa med prioritering	8
2.4	Vägplanering.....	9
2.4.1	A*	9
3	Problemformulering	10
3.1	Problembeskrivning	10
3.1.1	Delmål 1: Implementation	10
3.1.2	Delmål 2: Utvärdering	11
3.2	Metodbeskrivning.....	11
3.2.1	Metod för delmål 1: Implementation.....	11
3.2.2	Metod för delmål 2: Utvärdering	11
3.3	Metodreflektion	11
4	Genomförande	13
4.1	Förstudie.....	13
4.2	Progressionsexempel: modellering	13
5	Utvärdering.....	14
5.1	Presentation av undersökning.....	14
5.2	Analys.....	14
5.3	Slutsatser.....	14
6	Avslutande diskussion.....	15
6.1	Sammanfattning.....	15
6.2	Diskussion	15
6.3	Framtida arbete	15
	Referenser	16

1 Introduktion

Artificiell intelligens (AI) har alltid varit ett betydande område inom datalogi och det finns fortfarande mycket mer att lära inom området. Inom spel har AI ökat väldigt snabbt under de senaste åren och det läggs större krav på spel och dess AI i den aspekten att agenter ska bete sig trovärdigt. Söktekniker för vägplanering hos agenter är den AI som är vanligast inom spel. En vanlig vägplaneringsalgoritm som används på agenter i spel med statiska hinder är A* (A-stjärna). Utöver en vägplaneringsalgoritm krävs det någon sorts navigering hos agenterna i spelet för att de ska kunna röra sig efter den planerade vägen.

I takt att spel blir större blir även dess AI mer komplex. Hantering och navigering hos stora grupper av agenter i en spelmiljö är ett exempel på detta. Realtidstrategi-genren är en genre som behöver handskas med detta problem men det finns i andra genrer också. Exempelvis i ett FPS där civila människor springer och söker skydd.

Ett sätt att navigera agenter i en miljö är med hjälp av så kallade styrbeteenden. När ett styrbeteende appliceras på en agent kommer den agenten att kunna agera och ta egna beslut. Dessa beslut grundas på den informationen som ges från miljön runtomkring. De två tekniker för att navigera grupper av agenter som tas upp i detta arbete är flödesfälts- och vägföljnings-beteende. De två teknikerna används i kombination med ett antal andra styrbeteenden som till exempel hanterar kollision mellan agenterna och andra objekt i världen.

2 Bakgrund

2.1 Artificiell intelligens

Artificiell intelligens är konsten att skapa maskiner som utför uppgifter som kräver intelligens när de utförs av människor (Kurzweil, 1990). Artificiell intelligens handlar om att få datorer att utföra uppgifter och handlingar som människor och djur är kapabla att göra (Millington & Funge, 2009). Det är möjligt att programmera en dator att utföra uppgifter som är omöjliga för en människa (eller en grupp människor) att lösa under en rimlig tid. Exempelvis: sökning, aritmetiska problem, med mera.

Det är dock ett flertal uppgifter som datorer är dåliga på att utföra, som människor finner triviala: bestämma vad som ska göras närmast, känna igen ansikten och vara kreativa är endast några få exempel. Det är just detta som AI-området utforskar genom att undersöka vilka algoritmer det är som krävs för att få fram dessa egenskaper hos datorer.

2.1.1 Traditionell AI

Det traditionella AI-området är uppdelat i två, mindre, områden: stark AI och svag AI. Stark AI eftersträvar att skapa ett beteende som efterliknar människors tankeprocess, medan forskningen inom svag AI applicerar AI-teknologier på resultatet av verkliga problem. Buckland menar, i boken *Programming Game AI by Example* (2005), att dessa två subområden tenderar att fokusera på att lösa ett problem på ett sätt som tar mindre hänsyn till hårdvara eller tidsbegränsningar. Exempelvis kan en AI-forskare låta en simulation exekveras i timmar, dagar, eller veckor så länge det ger ett lyckat resultat som kan diskuteras i en artikel.

2.1.2 Spel-AI

Artificiell intelligens har alltid funnits inom datorspel, men *Pacman* (Namco, 1980) var det första spelet med en relativt avancerad AI. Fienderna rörde sig precis som spelaren och gav känslan av att det var riktiga människor som styrde dem (Millington & Funge, 2009).

En väldigt stor andel av de spel som finns idag har någon sorts AI implementerad. Oavsett om det är en hund som rör sig mellan två olika rum i ett hus, eller om det är en mer avancerad NPC (Non-Player Character) i ett rollspel som rör sig runt i en by, har båda agenterna ett sätt att navigera sig genom den miljö de befinner sig i. Inom datorspel är navigering och rörelse av agenter ett vanligt problem, oftast inom spel där en grupp av agenter ska navigeras tätt intill varandra. Tätt navigerande agenter är väldigt vanligt inom RTS (realtidspel) så som *Starcraft 2: Wings of Liberty* (Blizzard Entertainment, 2010) och *Warcraft 3: Reign of Chaos* (Blizzard Entertainment, 2002).

2.1.3 Autonom agent

En agent är någonting som kan uppfatta miljön runtomkring sig och agera utifrån den informationen. En agents beteenden beskrivs av agentfunktioner som mappar en given uppfattning till en mekanism (Russell & Norvig, 2009).

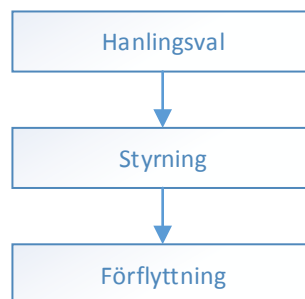
En autonom agent i ett datorspel är en enhet i en spelvärld som, till viss del, har egenskapen att improvisera sina beslut (Reynolds, 1999). Buckland definierar en autonom agent på följande sätt:

An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.

Buckland, 2005, s.85

Reynolds beskriver i *Steering Behaviors For Autonomous Characters* (1999) att ett beteende hos en autonom agent kan delas upp i flera lager för att lättare förstå det. Dessa lager är: handlingsval, styrning, och förflyttning (Figur 1).

- Handlingsval: Lagret som bestämmer vilket mål agenten har. Till exempel: "gå hit".
- Styrning: Lagret som ansvarar för att navigera agenten korrekt för att tillgodose målen som gavs från det tidigare lagret. Detta uppfylls genom att applicera styrbeteenden hos agenten för att producera en styrkraft som beskriver hur agenten ska röra sig.
- Förflyttning: Det lager som ansvarar för en agents förflyttning. Detta lager konverterar kontrollsignaler från styrningslagret till rörelse av agentens kropp.



Figur 1 Ett beteendes lager

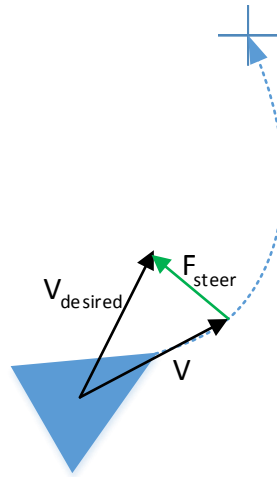
Ett exempel på en autonom agents beteende, i ett realtidstrategispel, är att spelaren ger order åt en autonom agent att röra sig till en ny position och den autonoma agenten navigerar sig själv genom miljön. Ett annat exempel är att ha två autonoma agenter, en råtta och en katt. Katten rör sig runt i en miljö medan musen sitter och äter. När musen ser att katten närmar sig flyr den från katten, samtidigt som katten börjar jaga musen. Alla dessa beslut görs utan någon översyn av en programmerare eller spelare.

2.2 Styrbeteende

Ett beteende som appliceras för att producera en styrkraft hos en agent, kallas för ett styrbeteende. Det finns en mängd olika styrbeteenden som producerar en styrkraft på olika sätt. Flera av styrbeteendena kan kombineras för att styra den autonoma agenten på ett mer komplext och naturligt sätt. Dessa styrbeteenden presenteras av Reynolds i hans artikel: *Steering Behaviors For Autonomous Characters* (1999).

2.2.1 Sök

Sök är ett styrbeteende som används för att styra en agent mot en specificerad position (P_{goal}), i global rymd. Beräkningen för detta styrbeteende är relativt enkel, först beräknas en önskad hastighet ($V_{desired}$) genom Formel 1. Med den önskade hastigheten går det att beräkna vilken styrkraft (F_{steer}) som ska appliceras på agenten med Formel 2.



Figur 2 Sökbeteendet

$$V_{desired} = normalized(P_{goal} - P) * V_{max}$$

Formel 1 Beräkningen för den önskade hastighet

$$F_{steer} = V_{desired} - V$$

Formel 2 Beräkningen för styrkraften

2.2.2 Ankomst

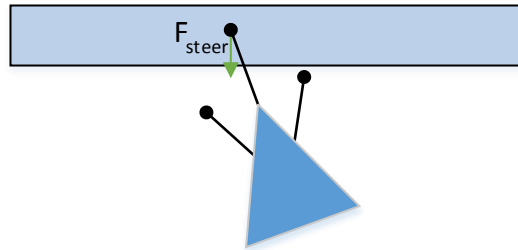
Styrebeteendet ankomst är identiskt till sök, så länge agenten är långt ifrån den specificerade målpositionen. Skillnaden från sökbeteenden, som styr sig mot målet, är att beteendet får karaktären att sakta ner när den närmare sig målet. Sträckan som definierar när agenten ska sakta ner går att förändra.



Figur 3 Ankomstbeteendet

2.2.3 Vägundvikande

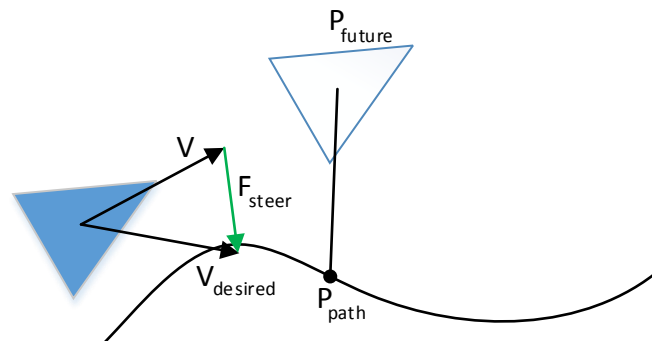
Vägundvikande ger en agent förmågan att styra ifrån potentiella kollisioner med väggar i en miljö. En vägg är ett linjesegment, i 3D en polygon. Detta görs med hjälp av avkännarpunkter för att se om en avkännarpunkt korsar en vägg. Om en avkännare korsar en vägg beräknas en styrkraft genom att beräkna hur mycket avkännaren har penetrerat väggen och sedan skapa en kraft av den penetrerade längden i väggens normalriktning.



Figur 4 Vägundvikelsebeteendet

2.2.4 Vägföljning

Vägföljningsbeteendet gör det möjligt för en agent att styras längs en serie av positioner som formar en väg. Agenten tittar, med hjälp av sin nuvarande position och hastighet, vilken position han kommer att ha om x sekunder (P_{future}). Från P_{future} beräknas den närmsta punkten på vägen (P_{path}). Ett sökbeteende appliceras sedan på agenten, med P_{path} som målposition. Om P_{path} skulle vara den sista punkten på vägen byts sökbeteendet ut mot ankomstbeteendet.

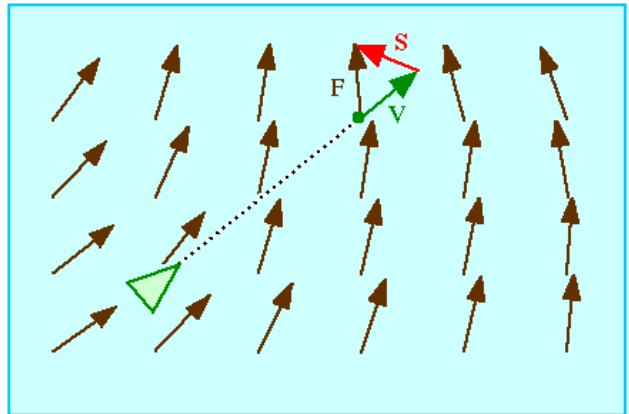


Figur 5 Vägföljningsbeteendet

2.2.5 Flödesfält

Styrbeteendet flödesfält är ett beteende för navigering av agenter, och kan användas som ett alternativ till vägföljningsbeteendet. Ett flödesfält är ett, godtyckligt stort, rutnät, där varje cell i rutnätet innehåller en riktningsektor. Denna riktningsektor representerar vilken styrkraft som ska appliceras på en agent när den befinner sig i cellen. Cellernas riktningsektorer kan vara statiska, men de kan också uppdateras dynamiskt. Dynamisk uppdatering av flödesfältet är fördelaktigt i spel där hinder för agenter förändras i realtid. Exempelvis realtidstrategispel, där agenterna inte får styras in i varandra och deras position alltid förändras.

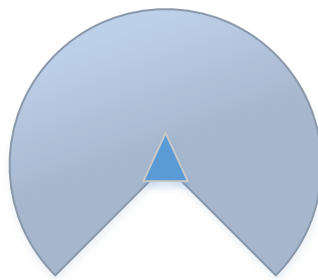
Under senare år har det skrivits ett antal artiklar där man tar upp denna typ av styrbeteende för att simulera stora grupper av agenter. Några exempel är *Continuum Crowds* (Treuille et al., 2006), använder flödesfält för att simulera en stadsmiljö med människor och bilar, *Directing Crowd Simulations Using Navigation Fields* (Patil et al., 2011), använder flödesfält för att simulera folkmassor i trånga utrymmen. Graham Pentheny (2013) diskuterar även om möjligheterna att använda flödesfält i spel där stora grupper av agenter måste förflytta sig i en miljö. Ett exempel på spel som använder flödesfält för att hantera grupper av agenter är *Planetary Annihilation* (Uber Entertainment, 2014) som är ett realtidstrategispel.



Figur 6 Flödesfältsbeteendet (Reynolds, 1999)

2.2.6 Flockbeteende

I *Flocks, Herds, and Schools: A Distributed Behavioral Model* (1987) beskriver Reynolds tre olika styrbeteenden som alla samarbetar för att skapa ett flockbeteende hos grupper av agenter. Dessa tre styrbeteenden är: Separation, sammanhållning, och formering. Styrbeteendena appliceras endast på en agent beroende på de agenter som befinner sig inom sitt närområde. Närområdet definieras med hur långt och brett en agents synfält är.



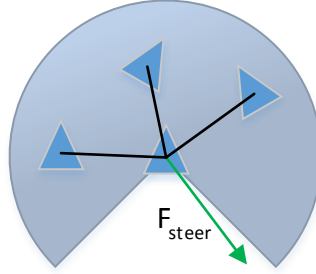
Figur 7 En agents närområde

2.2.7 Separation

Separationsbeteendet skapar en kraft som styr en agent ifrån andra agenter inom sitt närområde. När detta beteende appliceras på en mängd agenter kommer dom sprida ut sig, och försöka maximera längden från varandra. Kraften (F_{steer}) beräknas genom att beräkna en riktningsvektor mot alla agenter i närområdet. Riktningsvektorerna normaliseras och adderas sedan, se Formel 3. Detta beteende kan användas för att hindra agenter att tränga ihop sig.

$$F_{steer} = \sum_{i=1}^n normalized(P - P_i)$$

Formel 3 Beräkningen för styrkraften hos separationsebeteendet



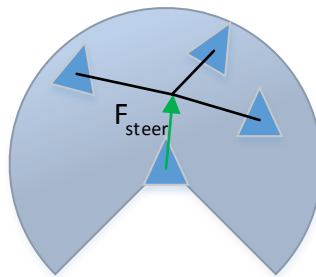
Figur 8 Separationsbeteendet

2.2.8 Sammanhållning

Sammanhållningsbeteendet ger en agent förmågan att närma, och gruppera, sig med andra agenter i närområdet. Styrkraften (F_{steer}) kan beräknas genom att beräkna medelpositionen hos de närliggande agenterna. Styrkraften kan sedan appliceras i riktningen från agenten och medelpositionen, se Formel 4.

$$F_{steer} = \frac{\sum_{i=1}^n P_i}{n} - P$$

Formel 4 Beräkningen för styrkraften hos sammahållningsbeteendet



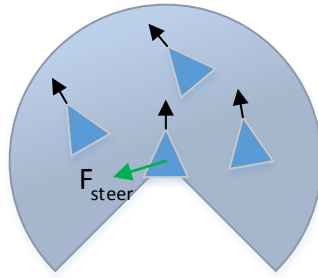
Figur 9 Sammanhållningsbeteendet

2.2.9 Formering

Formeringsbeteendet ger en agent förmågan att röra sig i samma riktning och hastighet som agenter i närområdet. Kraften (F_{steer}) kan beräknas genom att först beräkna medelhastigheten hos alla närliggande agenter och sedan subtrahera denna medelhastighet med agentens egen hastighet, se Formel 5.

$$F_{steer} = \frac{\sum_{i=1}^n V_i}{n} - V$$

Formel 5 Beräkningen för styrkraften hos formeringsbeteendet



Figur 10 Formeringsbeteendet

2.3 Beräkningsmodell för kombination av styrbeteenden

Det är sällan som en autonom agent endast styrs av ett enda styrbeteende, utan det är oftast en samling styrbeteenden som styr en autonom agents beteenden. Det krävs till exempel tre styrbeteenden för att åstadkomma ett flockbeteende hos agenter; separation, sammanhållning, och formering. Vill man till exempel att agenter ska söka sig till ett mål och samtidigt undvika kollision med andra agenter kan en lösning vara att kombinera sök- och separationsbeteendena.

Grunden bakom alla beräkningsmodeller är att addera ihop alla styrkrafter som produceras av styrbeteendena. Denna summa är den totala styrkraften som appliceras på agenten. Denna styrkraft får dock inte vara större än en specificerad maxkraft hos agenten. Två beräkningsmodeller beskrivs kort av Reynold i *Steering Behaviors For Autonomous Characters* (1999) medan Buckland (2005) går in på en mer detaljerad nivå när han beskriver dem.

2.3.1 Viktad trunkerad summa

Det enklaste sättet att kombinera styrbeteenden är genom att multiplicera varje styrbeteende med en vikt för att sedan trunkera resultatet med maxkraften hos agenten. Med denna beräkningsmodell medföljer dock några nackdelar (Buckland, 2005). Det första problemet är på grund av att varje aktivt styrbeteenden beräknas varje tidssteg, så är den väldigt ineffektiv att utföra. En annan nackdel är att det inte är trivialt att tilldela vikterna till alla aktiva styrbeteenden.

2.3.2 Viktad trunkerad summa med prioritering

Denna beräkningsmodell är väldigt lik den förra modellen, men den har två egenskaper som skiljer dem åt. Den första egenskapen är att varje styrbeteende har en prioritet som bestämmer hur viktig den är. Ett styrbeteende med en hög prioritering kommer att uppdateras före de som har låg prioritering. Exempelvis kan det vara viktigare att uppdatera att uppdatera separationsbeteendet, istället för sökbeteendet, hos en grupp agenter som inte ska kollidera med varandra.

Den andra egenskapen som skiljer denna modell ifrån den tidigare är att om den totala styrkraften hos en agent överskrider dess maximala tillåtna styrkraft kommer de resterande styrbeteendena, som har en lägre prioritering, inte att uppdateras. Det innebär att modellen ibland inte uppdaterar vissa styrbeteenden.

2.4 Vägplanering

Den vanligaste metoden för att få agenter att navigera sig igenom en miljö, i ett datorspel, är genom vägplanering. Den huvudsakliga uppgiften som vägplaneringen står för, är att hitta den kortaste vägen mellan två definierade punkter. Vägplaneringen tar även hänsyn till statiska objekt i miljön, såsom byggnader, berg osv., när den kortaste vägen mellan punkterna beräknas. A* (Hart et al., 1968) är en av de vanligaste algoritmerna för att beräkna den kortaste vägen.

2.4.1 A*

A*, även kallad A-stjärna, är en sökalgoritm som först framställdes av Peter Hart, Nils Nilsson, och Betram Raphael (1968). Algoritmen är en förbättring av Dijkstras (1959) algoritm och det som främst skiljer de två algoritmerna ifrån varandra, är att A* använder sig av heuristik för att hitta den kortaste vägen hos en graf. Heuristik innebär att algoritmen gör en uppskattning på vad avståndet från den nuvarande positionen till målpositionen. Funktionen för att beräkna det heuristiska värdet varierar från problem till problem och alla medför för- och nackdelar.

Anledningen till att A* är en vanlig algoritm för vägplanering inom dataspel är på grund av två egenskaper den har. Den första egenskapen är att algoritmen är komplett, vilket innebär att om det finns en väg till målet kommer den vägen att hittas. Den andra egenskapen är att algoritmen alltid kommer att hitta den mest optimala vägen, om det heuristiska värdet inte överskrider den verkliga kostnaden (Hart et al., 1968).

3 Problemformulering

Denna del av examensarbetet är uppdelat i två rubriker: Problembeskrivning och Metodbeskrivning. Problembeskrivning redovisar det problem som examensarbetet är baserat på. Metodbeskrivning redovisar hur examensarbetets frågeställning ska besvaras, undersökas, och utvärderas.

3.1 Problembeskrivning

Detta arbete har som syfte att jämföra olika tekniker för navigering hos grupper av autonoma agenter inom realtidstrategispel. Examensarbetet kommer att fokusera på två olika tekniker för navigeringen: Flödesfältsnavigering och vägföljningsnavigering. Båda teknikerna kommer att slås ihop med andra styrbeteenden när de utvärderas. Detta kommer att ske med hjälp av en beräkningsmodell. Detta är främst applicerbart hos spel som går inom genren realtidstrategispel men kan även användas inom andra spelgenrer, som till exempel rollspel där icke spelarstyrda karaktärer ska navigera sig i en miljö.

Det kommer att utföras experiment på grupper av autonoma agenter och storleken på dessa grupper kommer att variera från ett litet antal till ett hundratal agenter mellan de olika testfallen. Eftersom att miljöer i spel kan variera kommer testfallen att köras på ett flertal olika miljöer som lägger fokus på vissa aspekter, som till exempel trånga och fria utrymmen, korridorer, och dynamiska miljöer. Den frågeställning som arbetet kommer försöka besvara är:

- Hur jämför sig flödesfältnavigering gentemot vägföljningsnavigering hos grupper av autonoma agenter med avseende på tidseffektivitet?

För att det ska vara möjligt att besvara frågeställningen kommer en applikation skapas. Det ska vara möjligt att förändra vissa värden i applikation, som till exempel rutnätets densitet för navigering med flödesfält. Det ska även vara möjligt att välja en miljö som testerna ska utföras på.

I takten med att datorspel blir större och mer avancerade kommer antalet agenter i datorspel och komplexiteten på miljöer att öka. Detta kommer att ha en påverkan på hur effektiv navigeringen av agenterna är, men också hur estetiskt tilldragande de rör sig. Det är därför intressant att utföra tester för att se vilka navigeringstekniker som fungerar bäst under olika förhållanden.

Arbetets genomförande är indelat i två delsteg. Det första steget är att implementera en applikation vars syfte är att testa de två teknikerna, som arbetet ska utvärdera, i ett antal miljöer. Det andra steget är att göra utvärderingar på teknikerna med hjälp av den applikation som tidigare implementerats.

3.1.1 Delmål 1: Implementation

Avsikten med detta delmål är att implementera den applikation som kommer att användas för att utvärdera navigationsteknikerna. Därmed behöver alla styrbeteenden och beräkningsmodeller som ska användas, för utvärderingen, implementeras i applikationen. Det ska vara möjligt att skapa olika utfall för att enkelt kunna analysera teknikerna. Med olika utfall menas vilken miljö som ska användas, hur många agenter som ska navigera genom vald miljö, och vilken teknik de ska använda.

3.1.2 Delmål 2: Utvärdering

Detta delmål har som syfte att utvärdera navigeringsteknikerna. Det är i detta steg testfallen skapas med hjälp av applikation som implementerats i det tidigare delmålet. Vad som skiljer testfallen ifrån varandra är främst hur dess miljö ser ut, men även antalet agenter som navigerar genom miljön.

Den egenskap som arbetet kommer att utvärdera hos de två teknikerna är deras effektivitet. Den operationella definitionen av effektivitet i detta arbete är tidseffektiviteten och navigeringstiden hos teknikerna. Med tidseffektivitet menas antalet sekunder det tar för tekniken att hitta en väg för agenterna. Navigeringstiden är en mätning på hur lång tid det tar för agenterna att nå sin måldestination.

3.2 Metodbeskrivning

3.2.1 Metod för delmål 1: Implementation

Metoden för detta delmål är att implementera en applikation vars uppgift är att kunna besvara den frågeställning som arbetet har. Den mest vitala delen i detta delmål är implementationen av flödesfält- och vägföljningsnavigering. Styrbeteendena, och den beräkningsmodell, som agenterna kräver för att navigera sig i miljön måste också implementeras. Det krävs även att applikationen har en implementation av sökalgoritmen A^* , för att kunna hitta en väg som vägföljningsbeteendet kan använda.

3.2.2 Metod för delmål 2: Utvärdering

Metoden för denna del är att utvärdera de testfall som skapats i applikationen från tidigare del. För att göra det möjligt att utvärdera egenskapen av navigering kommer det att användas två mått. Navigeringen går ut på att en grupp agenter ska ta sig från en punkt till en annan i en arbiträr miljö. Dessa två mått är:

- Hur många sekunder tar det att generera en väg för agenterna?
- Hur många sekunder tar det för agenterna att ta sig till målpunkten?

För att utvärdera dessa mått hos teknikerna kommer ett flertal testfall utföras på de två teknikerna, tillsammans med styrbeteendena och en beräkningsmodell. För att utvärderingen ska bli så tydlig som möjligt kommer måtten hos teknikerna att jämföras och analyseras för att få fram vilken av teknikerna som skulle kunna lämpa sig bäst för navigering hos grupper av agenter.

Den motivation som ligger bakom användandet av dessa mått för att mäta tidseffektiviteten hos de två styrbeteendestekniker för navigering av autonoma agenter kommer mestadels från eget omdöme och till viss från Kotushevski (2010) mastersavhandling.

3.3 Metodreflektion

Det första delmålets metod finns det inte att mycket att reflektera över i och med att det är tvunget att en applikation implementeras för att det ska vara möjligt att få fram ett resultat som besvarar arbetets frågeställning.

Det går däremot att diskutera över den metod som valts för det andra delmålet. Som alternativ till att jämföra teknikernas tidseffektivitet baserat ett antal definierade mått hade

det hade varit möjligt att utvärdera resultaten genom att ha testpersoner som får navigera med agenterna för att sedan ha en enkätundersökning med dessa personer. Anledningen till valet som gjordes är för att resultaten från testfallen ska vara mer objektiv. Att göra tester och enkätundersökningar på personer som får testa teknikerna kan vara en nackdel i och med att AI och speciellt vägplanering kan tyckas vara väldigt subjektiv. Något som också är värt att ha i åtanke om testpersoner och enkätundersökningar används som metod för det andra delmålet är att om det skulle hittas ett problem i någon navigeringsteknikernas implementation efter att undersökningarna har utförts är resultaten från testfallen i stort sett helt inaktuella.

4 Genomförande

4.1 Förstudie

4.2 Progressionsexempel: modellering

5 Utvärdering

5.1 Presentation av undersökning

5.2 Analys

5.3 Slutsatser

6 Avslutande diskussion

6.1 Sammanfattning

6.2 Diskussion

6.3 Framtida arbete

Referenser

- Blizzard Entertainment (2010). *Starcraft 2: Wings of Liberty* (Version: 1.0) [Datorprogram]. Blizzard Entertainment.
- Blizzard Entertainment (2002). *Warcraft III: Reign of Chaos* (Version: 1.0) [Datorprogram]. Blizzard Entertainment.
- Buckland, M. (2005). *Programming Game AI by Example*.
- Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*. (<http://csl.mendeley.com/styles/252469921/harvard-skovde-university-2>). s. 269–271.
- Hart, P., Nilsson, N.J. & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*. s. 100–107.
- Kotushevski, G. (2010). *Intelligent driver agent model for autonomous navigation in a computer simulated vehicular traffic network*. Massey University.
- Kurzweil, R. (1990). *The age of intelligent machines*.
- Millington, I. & Funge, J. (2009). *Artificial Intelligence for Games, Second Edition*.
- Namco (1980). *Pacman* (Version: 1.0) [Datorprogram]. Namco.
- Patil, S., Berg, J. van den, Curtis, S., Lin, M.C. & Manocha, D. (2011). Directing crowd simulations using navigation fields. *IEEE transactions on visualization and computer graphics*. s. 244–54.
- Pentheney, G. (2013). *GDC - The Next Vector: Improvements in AI Steering Behaviors*. 2013. Tillgänglig på Internet: <http://www.gdcvault.com/play/1018230/The-Next-Vector-Improvements-in>. [Hämtad 30 January 2015].
- Reynolds, C. (1999). *Steering Behaviors For Autonomous Characters*.
- Reynolds, C.W. (1987). Flocks, herds and schools: A distributed behavioral model. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH '87*. 1 August 1987, New York, New York, USA: ACM Press, s. 25–34.
- Russell, S. & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*.
- Treuille, A., Cooper, S. & Popović, Z. (2006). Continuum crowds. *ACM Transactions on Graphics*. s. 1160.
- Uber Entertainment (2014). *Planetary Annihilation* (Version: 1.0) [Datorprogram]. Uber Entertainment.