Examensarbete

HÖGSKOLAN I SKÖVDE

MEDHJÄLPAR-AI I SPEL

Skapande av en dynamiskt anpassningsbar AI i spel med hjälp av en genetisk algoritm

Examensarbete inom huvudområdet Datalogi
Grundnivå 30 högskolepoäng
Vårtermin 2012

Louise Wahlstrand

Handledare: Mikael Thieme
Examinator: Sanny Syberfeldt

Sammanfattning

Medhjälpar-AI blir allt mer vanligt i dagens spel och främst används de som substitut till en mänsklig spelare i spel med olika typer av samarbetslägen. Det finns med andra ord ett behov att utforska detta område, både när det gäller olika AI-tekniker och metoder för förbättring av hur medhjälpar-AI:n upplevs. Genetiska algoritmer är en AI-teknik som simulerar evolution, vilket är användbart i flera områden. Det här arbetet undersöker om genetiska algoritmer kan fungera som basis till en medhjälpar-AI som anpassar sig efter spelarens prestation i realtid.

Experimentmiljön som används är ett enkelt actionspel, där en spelare tillsammans med medhjälpare ska samla poäng. Beroende på en medhjälpares poäng i förhållande till spelaren, ändrar medhjälparen skicklighetsnivå dynamiskt under spelets gång.

Utvärderingen visar att genetiska algoritmer fungerar bra som utgångspunkt till ändamålet. Med andra ord visar arbetet att genetiska algoritmer har god potential att fungera bra som grundstruktur för mer avancerad medhjälpar-AI.

Nyckelord: Adaptiv AI, Genetiska algoritmer, Medhjälpar-AI, Realtidsevolution

Innehållsförteckning

1 Introduktion 1
2 Bakgrund 2
2.1 AI-tekniker 2
2.2 Evolutionära tekniker – En översikt 2
2.3 Genetiska algoritmer 3
2.3.1 Fitness-evaluering 4
2.3.2 Selektion 4
2.3.3 Överkorsning 5
2.3.4 Mutation 6
2.4 Samevolution 6
2.5 Relaterad forskning: Del 1 6
2.5.1 Spelet – Experimentmiljön 7
2.5.2 Monstrens AI 7
2.5.3 Metod 1: Realtidsevolution med användning av spelspecifik information 7
2.5.4 Metod 2: Realtidsevolution med användning av offline-evolverad data 8
2.5.5 Metod 3: Gedigen realtidsevolution 8
2.5.6 Metod 4: Hybrid 8
2.5.7 Resultat och slutsats 8
2.6 Relaterad forskning: Del 2 9
3 Problemformulering 10
3.1 Delmål 11
3.1.1 Delmål 1: Implementation 11
3.1.2 Delmål 2: Utvärdering 11
3.2 Metodbeskrivning 11
4 Implementation 14
4.1 Experimentmiljön 14
4.2 AI:n 15
4.2.1 Spelaren 15
4.2.2 Medhjälparna 16
4.2.3 Uppbyggnad och hantering av genom 16
4.3 Balansering 18
4.4 Koddesign 19
5 Analys 21
6 Slutsatser 25
6.1 Resultatsammanfattning 25
6.2 Diskussion 25
6.3 Framtida arbete 27

1 Introduktion

AI utgör idag en viktig del i många spel och är något som kan påverka spelupplevelsen väldigt mycket. AI i spel visar sig i flera former och det vanligaste kanske är som motstånd, men AI finns överallt i spel: byinvånarna i *The Legend of Zelda: Ocarina of Time* (Nintendo EAD, 1998), polisbilarna i *Grand Theft Auto IV* (Rockstar North, 2008), *Little Sisters* i *Bioshock* (Irrational Games, 2007) och truppmedlemmarna i *Gears of War 3* (Epic Games, 2011) är alla exempel på AI. Det sistnämnda är ett exempel på medhjälpar-AI; något som idag är vanligt i spel.

En medhjälpar-AI kan till exempel agera substitut till en mänsklig spelare i spel tänkt för samarbete mellan två eller fler spelare, där exempelvis vissa pusselmoment kräver samarbete (som i *Ratchet & Clank: All 4 One* (2011) av Insomniac Games). Ett annat exempel är om det på grund av storyn är nödvändigt för spelaren att ha en eller flera medhjälpare, som i *Mass Effect 2* (BioWare, 2010). Även i lagspel finns behovet av en AI som kan fungera som ersättare till en spelare ifall en sådan saknas, något som tillämpas i till exempel *Gears of War 3* (Epic Games, 2011).

Medhjälpar-AI är ett intressant ämne som gör det önskvärt att låta AI:n förhålla sig till spelarens förmåga. En AI som anpassar sig efter spelaren med målet att varken prestera bättre eller mycket sämre, skulle troligtvis upplevas som en relativt välbalanserad AI oavsett skicklighetsnivå på spelaren den förhåller sig till.

Enligt Bourg och Seemann (2004) är genetiska algoritmer en bra utgångspunkt för att uppnå en balanserad AI. Genetiska algoritmer är en huvudteknik inom evolutionär AI som grundar på Darwins princip om att det är större sannolikhet att lämpliga individer blir utvalda att producera en avkomma till nästa generation (Hoffmann, 1998) och på det sättet föra sina gener vidare. På samma sätt har en AI som presterar bra (sett till problemet) större chans att genom sina "gener" föra delar av sitt beteende vidare till nästa generation.

Förhållandet mellan medhjälpar-AI:n och spelaren är också något som kan liknas vid samevolution, eftersom evolutionen som medhjälpar-AI:n genomgår delvis styrs av spelarens skicklighet. Liknelsen härstammar från biologin, där samevolution enligt Nationalencyklopedin (2012) innebär att en eller fler arter påverkar varandra på ett sätt att de genomgår evolutionära förändringar.

Det här projektet undersöker hur väl genetiska algoritmer kan utnyttjas som grundstomme till en dynamiskt anpassningsbar medhjälpar-AI, genom att utveckla och utvärdera sådan AI i en experimentduglig spelmiljö.

Förutom sammanfattning och denna introduktion, som utgör det första respektive andra kapitlet, är rapporten indelad i fem kapitel. Kapitel tre går igenom bakgrunden, det vill säga AI-tekniker relevanta för problemet samt relaterad forskning. Vidare beskriver kapitel fyra problemet samt delmål och metod för projektet. Det femte kapitlet är en genomgång av experimentmiljön och dess implementation. Kapitel sex är en utvärdering av arbetet som följs av en slutsats med resultatsammanfattning, diskussion och framtida arbete i det sjunde och sista kapitlet.

2 Bakgrund

För att ge en helhetsförståelse av projektet tar det här kapitlet inledningsvis upp några av de vanligaste AI-teknikerna, följt av en genomgång av evolutionära tekniker i allmänhet. Därefter fördjupning i genetiska algoritmer och delar av evolutionsprocessen. Det sista avsnittet i kapitlet tar upp samevolution i kombination med relaterad forskning som vidare kopplas ihop med ämnet för projektet: medhjälpar-AI.

2.1 AI-tekniker

Enligt Bourg och Seemann (2004) delas AI vanligtvis in i två stora områden: deterministisk och icke-deterministisk AI. Deterministiska AI-tekniker är relativt snabba och lätta att implementera, testa och debugga. En nackdel är att sådan AI tenderar att bli förutsägbar i sitt beteende, vilket i sin tur kan förkorta spelets livslängd. Dessutom krävs det att programmeraren täcker alla möjliga scenarion med AI:n i dess implementation, vilket kan vara en nackdel. Ett exempel på deterministisk AI är spökena i Pac-Man (Namco, 1980) som är implementerade med hjälp av tillståndsmaskiner och har ett förutbestämt beteende.

Vidare skriver Bourg och Seemann (2004) att icke-deterministisk AI däremot främjar oförutsägbart och självutvecklande beteende. En sådan AI är dock svår att debugga eftersom den är just oförutsägbar, vilket förstås är en nackdel.

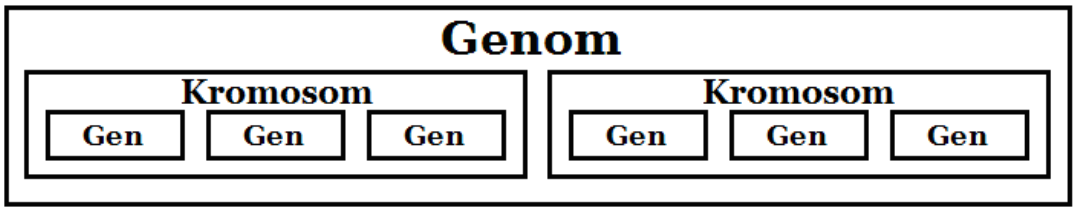
Icke-deterministiskt beteende kan implementeras med hjälp av bland annat ett neuralt nätverk eller en evolutionär teknik som exempelvis genetiska algoritmer. Vidare påpekar Bourg och Seemann (2004) att exempel på icke-deterministisk AI finns i det framgångsrika spelet Black & White (Lionhead Studios, 2001).

2.2 Evolutionära tekniker – En översikt

Enligt Mitchell (1998) växte evolutionära tekniker fram redan under 50- och 60-talet, då ett antal datorforskare studerade evolutionära system oberoende av varandra. Inspirerade av biologin var tanken att evolution kunde användas för att lösa tekniska optimeringsproblem. Många beräkningsproblem kräver nämligen att ett enormt antal möjliga lösningar genomsöks och enligt insatta forskare verkar mekanismerna inom evolution väl lämpade för det ändamålet (Mitchell, 1998). Ett problem där evolution tillämpas är i det välkända "The Traveling Salesman Problem" (Buckland, 2002), för att nämna ett exempel.

Evolution var, och är, en metod att hitta eventuella lösningar på ett givet problem genom att söka igenom ett stort antal möjligheter. Vidare skriver Mitchell (1998) att en uppsättning möjligheter inom biologin är en mängd av möjliga genetiska sekvenser, där de önskade "lösningarna" är starkt lämpade organismer som i sin miljö kan överleva och producera avkommor. På samma sätt fungerar evolution inom datavetenskap men där består mängden möjligheter och lösningar av data istället.

I och med parallellen till biologin adapterades även termer därifrån. Enligt Mitchell (1998) kallas en lösning till ett givet problem för genom, där ett genom består av ett antal kromosomer. En kromosom utgörs i sin tur av ett antal gener, där en gen motsvarar ett värde (till exempel ett heltal). Figur 1 visar förhållandet mellan dem:



Figur 1 Förhållandet mellan genom, kromosom och gen.

Genom att utföra genetikinspirerade operationer på kromosomerna, såsom selektion, mutation och överkorsning (som alla beskrivs närmare i kapitel 2.3), evolveras potentiella lösningar fram till ett givet problem. Dessa potentiella lösningar kallas normalt för kandidatlösningar. De individer som visar sig vara mest lämpade för problemet, är också de som har störst chans att föra sina gener (sin data) vidare till nästa generation. Enligt Buckland (2002) kallas måttet på hur framgångsrik en individ är för *fitness*; ju bättre fitness, desto större är sannolikheten att individen får producera avkomma.

Idag finns det tre huvudsakliga teknikerna inom evolutionär AI: genetiska algoritmer (GA), evolutionära strategier och genetisk programmering (Negnevitsky, 2004). Gemensamt för huvudteknikerna är att simulera evolution, där genetiska algoritmer är den mer generella tekniken av de tre. Enligt Negnevitsky (2004) kan evolutionära strategier och genetisk programmering ses som varianter på GA.

En skillnad mellan GA och evolutionära strategier är att den förstnämnda har överkorsning som huvudsaklig operation i evolutionsprocessen medan den sistnämnda fokuserar på mutation (Hoffmann, 1998). Genetisk programmering kan däremot ses som en utbyggnad på GA med målet att skapa ett program som löser ett givet problem, istället för att endast evolvera fram kandidatlösningar till problemet i form av kromosomer (Negnevitsky, 2004).

2.3 Genetiska algoritmer

I genetiska algoritmer används ofta binära strängar för att representera kromosomerna (Hoffmann, 1998; Mitchell, 1998). Denna sträng kallas även för den genetiska koden (Demasi & Cruz, 2002) och är den del som styr beteendet hos AI:n i det här arbetet. Mitchell (1998) påpekar också att de flesta applikationer med GA använder kandidatlösningar som består av en enda kromosom, vilket även är fallet i det här projektet: en medhjälpar-AI baseras på en kromosom där varje gen är ett binärt tal (0 eller 1). För att tydliggöra är alltså en kromosom och ett genom samma sak i det här projektet, eftersom ett genom endast består av en enda kromosom (en kandidatlösning). Det är även viktigt att påpeka att genetiska algoritmer i allmänhet inte garanterar någon lösning (Buckland, 2002).

Enligt Bourg och Seemann (2004) skulle evolutionsprocessen för en genetisk algoritm i spel kunna delas in i följande fyra delar (del två, tre och fyra tas upp noggrannare i efterkommande delkapitel):

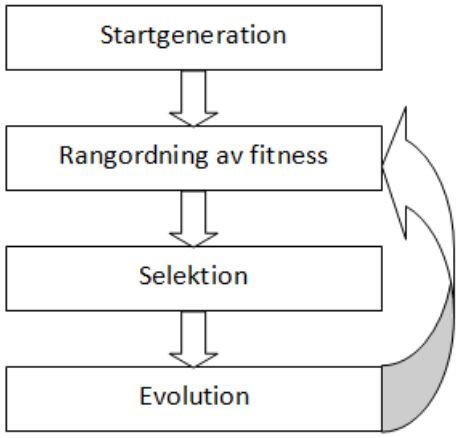
1. Förstageneration; varje genom i startgenerationen representerar en möjlig lösning till det givna problemet. En sådan lösning motsvarar i projektet beteendet hos AI:n. Ett sätt att skapa startgenerationen är att slumpa värdena i kromosomerna.

2. Rangordning av fitness; i det här steget evalueras fitness för varje genom.

3. Selektion; i det här steget väljs ett antal genom ut för producering av nästa generation.

4. Evolution; en ny generation tillkommer genom att utföra överkorsning och mutation på de utvalda genomen från föregående steg.

Steg 2-4 upprepas tills en ny generation av individer (genom) har skapats. Figur 2 ger en överblick av processen:



Figur 2 Evolutionsprocessen.

2.3.1 Fitness-evaluering

För att kunna avgöra hur väl ett genom fungerar som lösning givet ett visst problem, behöver någon typ av evaluering genomföras. Detta görs vanligtvis med hjälp av en fitness-funktion och vitsen med den är att rangordna genomen och på det viset avgöra vilka som är bäst lämpade att lösa problemet (Buckland, 2002).

För problemet i det här arbetet utgörs en medhjälpares fitness av hur många procent av spelarens poäng den har, vilket ger en indikation på hur framgångsrik medhjälparen är när det gäller att anpassa sig efter spelaren.

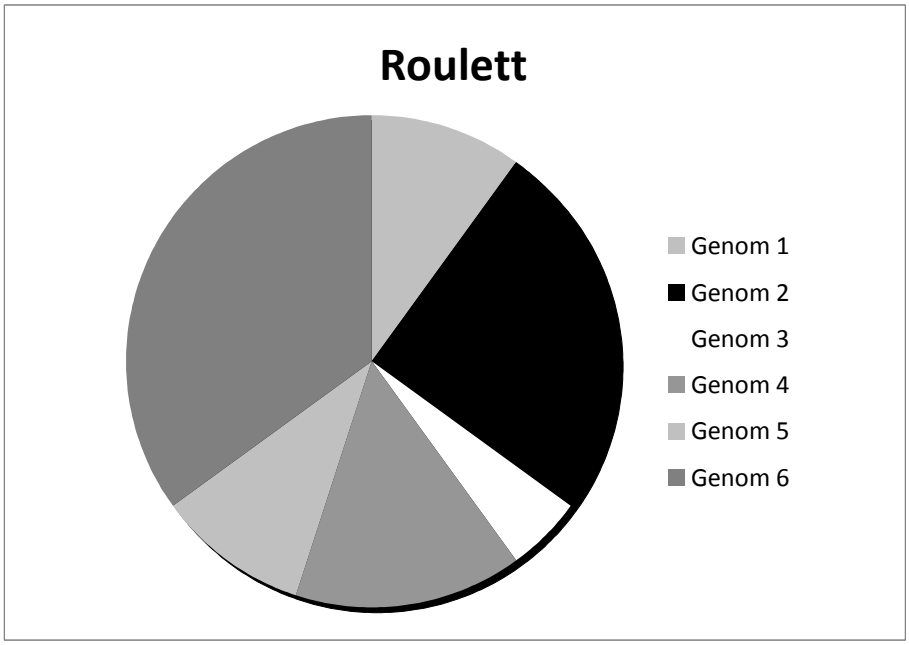
2.3.2 Selektion

I den biologiska världen används vanligtvis kromosomer från två individer för att producera avkomma men enligt Buckland (2002) finns det inget i spelutveckling som hindrar att ett godtyckligt antal genom väljs ut som "föräldrar". Med andra ord kan gener från exempelvis fem genom bilda ett nytt genom. Även om ett genom har valts ut som förälder tas det inte bort, vilket innebär att genomet kan väljas om på nytt.

Det finns olika tekniker att genomföra selektion på och några av de vanligaste är elitism, roulett och turnering. Den förstnämnda, elitism, används ofta ihop med andra selektionsmetoder och är ett sätt att behålla ett antal av de mest lämpade genomen till nästa generation (Mitchell, 1998).

I roulettselektion ges varje genom en andel som motsvarar dess fitness-kvot (se Figur 3) i förhållande till populationens totala fitness (Negnevitsky, 2004). Fitness-kvoten är det som avgör hur stor sannolikheten är att ett genom blir utvalt för överkorsning (som förklaras i nästa delavsnitt). Ett genom med högre fitness har alltså större chans att bli utvalt än ett genom med lägre, då dess andel är större. Processen upprepas tills en ny generation tillkommit. En nackdel med roulettselektion är dock risken att endast de genom med lägre

fitness väljs ut, därför kan en god idé vara att kombinera roulett med elitism (Buckland, 2002).



Figur 3 Roulettselektion; varje bit motsvarar fitness-kvoten hos ett genom.

Buckland (2002) skriver att i turneringsselektion är det första steget att välja ut ett godtyckligt antal genom slumpmässigt. Därefter väljs det med bäst fitness ut och "går vidare" till överkorsning. Likt roulettmetoden upprepas denna process tills en ny population är framtagen. En nackdel med turneringsselektion är precis som med roulettselektion att det finns risk att endast de genom med lägre fitness blir utvalda. Risken går dock att minimera genom att öka antalet genom som väljs ut slumpmässigt inför turneringen.

I det här projektet är selektionsdelen en specialgjord sådan eftersom tanken är att AI:n ska anpassa sig efter spelaren. Med inspiration av Demasi och Cruz (2002), kommer AI:n att paras ihop med något av två fördefinierade genom varje gång det är aktuellt för en evolution att genomföras. En noggrannare genomgång av detta finns under relaterad forskning i kapitel 2.5.3.

2.3.3 Överkorsning

De genom som valts ut till överkorsning kommer i det här steget att producera ett nytt genom (kromosom) genom en kombination av deras gener. Vanligtvis används en korsningsfrekvens för att avgöra om överkorsning ska genomföras över huvud taget. Skulle överkorsning skippas replikeras de utvalda genomen istället (Mitchell, 1998; Negnevitsky, 2004).

Överkorsning kan ske på många olika sätt men en av de enklare metoderna är enkelpunktskorsning. Det innebär inledningsvis att en position (korspunkt) slumpas fram. Därefter växlas ändsekvenserna, alltså den sekvens av gener som ligger efter den framslumpade positionen, mellan de iblandade genomen för att skapa avkomman (Mitchell, 1998; Negnevitsky, 2004). Om korspunkten exempelvis är mellan den första och andra genen, skulle en överkorsning mellan genomen **0101** och **1111** resultera i de två nya genomen **0111** respektive **1101**.

Buckland (2002) beskriver två andra metoder: tvåpunktskorsning och multipunktskorsning. Den förstnämnda har, som namnet antyder, två korspunkter. Segmentet som bildas mellan de slumpade korspunkterna är den delsekvens som växlas mellan genomen. Med korspunkter efter andra och femte genen, skulle genomen 00**100**111 och 10**011**101 resultera i avkommorna 00**011**111 respektive 10**100**101.

Multipunktskorsning innebär att ett godtyckligt antal gener på framslumpade positioner byts ut. Exempel: De gener markerade med fetstil i genomen **10**10**0**1 och **111**0**1** är de som växlas. Resultatet blir de nya genomen **11**10**0**1 respektive **10**11**0**1.

I det här projektet är det önskvärt med en mjuk övergång när medhjälpar-AI:n evolveras, då en alltför hastig förändring i AI:ns beteende gärna undviks. I ett försök att uppnå det används multipunktskorsning med ett lågt antal gener, eftersom det jämfört med andra metoder troligtvis minimerar risken till att ett sådant hastigt förändrande beteende hos AI:n, som kan uppfattas som överdrivet ologiskt, visar sig.

2.3.4 Mutation

Enligt Negnevitsky (2004) innebär mutation en förändring i generna hos ett genom – något som kan leda till ett bättre fitness men som oftast resulterar i ett sämre. Trots det är mutation betydelsefullt för att algoritmen inte ska hamna i ett lokalt optima, vilket sker ifall genomen i en population är nära på identiska. Är genomen för lika kommer ingen större förändring att ske i överkorsningssteget och därmed minskar chansen att hitta optimala lösningar på problemet. Mutation är med andra ord viktigt för att bevara mångfalden i populationen.

Mutation utförs genom att ändra gener på slumpmässiga positioner i genomet (Mitchell, 1998). Om en markerad gen har värdet 1 skulle den efter mutationen bli 0 i ett genom med binära värden. Mutation kan inträffa i varje gen men sannolikheten att den ska muteras är vanligtvis väldigt liten: mellan 0.001 och 0.01 är typiskt enligt Negnevitsky (2004).

I det här projektet innebär mutation att det är en viss sannolikhet att en enda gen ändras på slumpmässig plats i genomet, därför kommer sannolikheten istället vara mycket högre än 0.001-0.01 då endast en gen kan påverkas.

2.4 Samevolution

Samevolution inom biologin innebär att två eller flera arter påverkar varandra på ett sätt att de genomgår evolutionära förändringar (Nationalencyklopedin, 2012). Relationen mellan insekter och dem växter de pollinerar är ett exempel på samevolution. På liknande sätt fungerar relationen mellan spelaren och AI:n i det här arbetet, där AI:n i realtid ska sträva efter att, inom vissa gränser, förhålla sig till spelaren.

2.5 Relaterad forskning: Del 1

Artikeln *Online Coevolution for Action Games* (2002) av Pedro Demasi och Adriano J. de O. Cruz handlar om samevolution i realtid och fungerar som en utgångspunkt och inspirationskälla för det här projektet. Syftet med Demasis och Cruz (2002) arbete är att utforska samevolution genom applikationer som använder samevolutionära algoritmer. De fokuserar på detta i kombination med realtidsinteraktion och går i artikeln igenom olika metoder och strategier för realtidsevolution i ett actionspel.

2.5.1 Spelet – Experimentmiljön

Deras spel går ut på att en mänskligt styrd karaktär ska döda så många monster som möjligt. Spelaren dödar ett monster genom att träffa den med ett skott. Vidare fås ammunition via lådor med patroner som dyker upp på planen med jämna mellanrum, där varje låda ger 20 skott. Ett monster dödar spelaren genom att kollidera med den, vilket resulterar i ett liv mindre (spelaren har tre liv från början).

Monstren är datorstyrda och allt eftersom spelaren dödar dem ska motståndet öka (utan att antalet monster ökar; det ska alltid finnas 16 på spelplanen). Med andra ord är tanken att svårighetsgraden ska trappas upp gradvis genom att monstren blir intelligentare. Det som sker mellan spelaren och karaktären är alltså samevolution, för ju bättre spelaren presterar desto snabbare ökar motståndet till följd av att monstren utvecklas. För att uppfylla detta evolveras monstren under spelets gång, det vill säga i realtid (eller *online* som det även benämns som).

2.5.2 Monstrens AI

Monstrens genetiska kod är binär och evolutionen sker med hjälp av genetiska algoritmer. Det ska dock uppmärksammas att det inte är specifika monster som evolveras, utan själva monsterpopulationen. När ett monster har dött ersätts det alltså med ett nytt, "framavlat" monster.

Monsterpopulationen är indelad i två subtyper där hälften strävar efter ett visst beteende och den andra hälften efter ett annat. Den ena subtypens uppgift är främst att "vakta" ammunition och undvika skott medan den andra typen istället jagar spelaren samtidigt som den försöker undvika skott.

2.5.3 Metod 1: Realtidsevolution med användning av spelspecifik information

Den första metoden som Demasi och Cruz (2002) implementerar är vad de kallar *realtidsevolution med användning av spelspecifik information*. I den metoden används två fördefinierade monster (ett för varje subtyp) som ideal för evolverande monster att sträva efter. Dessa ideal benämns även som *targets*.

I det första tillvägagångssättet för metoden använder Demasi och Cruz (2002) Hammingavstånd för att styra evolutionen. Hammingavståndet där är antalet bitar (en bit är antingen en 1:a eller en 0:a) som skiljer sig mellan monstret och dess target. När ett monster har dödats beräknas Hammingavståndet mellan den och dess target, sedan ändras en av bitarna som skiljer sig (vilken avgörs av slumpen). På så sätt skapas ett monster som är lite mer lik sitt target än dess förälder. Skulle spelaren inte ha presterat tillräckligt bra kommer monstret dock att replikeras istället. Spelarens prestation beror bland annat på antalet skott som träffar.

Det andra tillvägagångssättet är att låta ett dödat monster korsas med ett av idealen, för att sedan låta avkomman ersätta det döda monstret. Likt föregående tillvägagångssätt sker detta endast om spelaren har presterat tillräckligt bra, annars korsas monstren med varandra där föräldrarna bestäms genom roulettselektion.

En fördel med den första metoden är avsaknaden av tunga beräkningar. Nackdelen är dock att den kräver fördefinierade targets, vilket inte alltid är helt trivialt. Ytterligare en fördel, som gäller tillvägagångssätt två, är att det finns potential till mjuka övergångar i evolutionen.

2.5.4 Metod 2: Realtidsevolution med användning av offline-evolverad data

I andra metoden designas targets istället med hjälp av offline-evolverad data. Huvudidén samt implementationen påminner mycket om första metodens. Agenterna (monstren) evolveras alltså offline och resultatet av de genererade agenterna medför många targets att välja mellan.

Tillvägagångssätten som föreslås i metod två är att antingen implementera en AI som styr spelaren eller låta ett stort antal testpersoner spela mot agenterna. Med det förstnämnda kan grovarbetet överlåtas till systemet som kan hålla igång i timmar, eller dagar, tills ett önskvärt resultat är uppnått. Med det sistnämnda kan mycket mångfald uppnås bland agenterna eftersom det produceras så många - något som kan vara väldigt användbart när targets ska väljas ut för evolution. Nackdelen är att det behövs en hel del resurser för att genomföra en sådan omfattande testning, då ett stort antal testpersoner är nödvändigt. Demasi och Cruz (2002) valde av den anledningen att endast genomföra det första tillvägagångssättet i den här metoden: implementera en AI-styrd spelare.

2.5.5 Metod 3: Gedigen realtidsevolution

Om det inte går att designa targets baserat på föregående metoder återstår gedigen online-evolution, vilket innebär att endast aktuell speldata används. Till detta ändamålet används en "pool" bestående av de agenter med högst fitness. Det finns en pool för varje subtyp. När en agent dör korsas den med någon från dess pool; roulettselektion avgör med vilken. Den med sämst fitness i poolen ersätts ifall den nya agenten har högre fitness. En nackdel med den här metoden är dock att evolutionsövergångarna kan bli "hackiga" i spel där agenter inte ersätts speciellt ofta.

2.5.6 Metod 4: Hybrid

Den fjärde och sista metoden är att kombinera den föregående med antingen den första eller andra metoden. Som exempel kan targets definieras genom metod 1 eller 2 och när agenterna nått sin target-nivå går de över till gedigen online-evolution (metod 3).

Demasi och Cruz (2002) skriver att metoden inleder med ett introduktionsstadium, som de kallar det, där spelaren först måste slå agenter med grundläggande strategier. Andra stadiet skulle då fungera som ett avancerat läge där agenterna evolverar online och utvecklar mer komplexa strategier som är mer anpassade till spelarens spelsätt. Demasi och Cruz (2002) valde att kombinera metod 3 med första tillvägagångssättet i metod 2 (AI-styrd spelare).

Den sista metoden kan ge goda resultat ifall agenterna har grundläggande strategier som fungerar bra mot mindre erfarna spelare och det inte går att bestämma en bästa strategi för en agent. I vissa fall kanske det inte ens finns en bästa eftersom spelsätt skiljer sig spelare emellan: en strategi kan fungera bra mot en viss spelare men likväl kan den fungera dåligt mot en annan.

2.5.7 Resultat och slutsats

Genom dessa metoder var Demasis och Cruz (2002) förhoppning att monsterpopulationen gradvis skulle bli svårare att slå, vilket deras resultat också visar: generellt sett minskar tiden som spelaren lever för varje liv som förbrukas – alltså i takt med att agenterna evolveras. Tillvägagångssätt två i metod 1 skiljer sig dock lite från de andra då spelaren klarar sig längst på sitt andra liv (istället för första som är det önskade resultatet), vilket visar på en relativt långsam evolution.

Alla metoder skulle troligtvis ge intressanta resultat om tillämpade för evolution av en anpassningsbar medhjälpar-AI i spel. Det som ligger till grund för medhjälpar-AI:n i det här projektet är dock en hybrid av metod 1 och 2. En evolution med mjuka övergångar är önskvärd och Demasis och Cruz (2002) resultat visar att evolutionen i metod 1 är något långsammare jämfört med de andra metodernas. En långsammare evolution passar troligtvis bra till en medhjälpar-AI som ska anpassa sig efter spelaren, eftersom risken finns att den ofta kommer behöva växla mellan ett intelligent och ett mindre intelligent beteende. Ju långsammare evolution desto mindre "hopp" i AI:ns beteende, vilket förstås är en fördel. Samtidigt får evolutionen givetvis inte vara för långsam eftersom AI:n trots allt ska byta beteende. Då den genetiska koden inte är speciellt lång för medhjälparna i det här projektet är dock den risken liten.

Targets används även i det här projektet och dessa evolveras fram offline, vilket bygger på metod 2. Alltså används en kombination av den första och andra metoden.

2.6 Relaterad forskning: Del 2

I projektet som beskrivs i artikeln *AI for Dynamic Team-mate Adaption in Games* (2010) utvecklar Aswin Thomas Abraham och Kevin McGee en dynamisk medhjälpar-AI. Tillsammans med spelaren samarbetar medhjälpar-AI:n för att övervinna motståndet som, likt forskningen i kapitel 2.5, gradvis trappas upp. Syftet med projektet är bland annat att formalisera utvecklingen av en medhjälpar-AI med anpassningsbar svårighetsgrad.

Till skillnad från det här arbetet handlar projektet i artikeln mer om att komplettera spelaren; till exempel genom att anta en annan roll än spelaren. I spelet som Abraham och McGee (2010) utvecklat behöver spelaren och medhjälparen samarbeta för att fånga motståndaren: en skytt. Om spelaren då antar rollen som "jagare" anpassar medhjälpar-AI:n sig efter spelaren och antar rollen som avledare/"lockbete". Detta skiljer sig en del från det här arbetet där medhjälparen istället anpassar sig genom att jämföra sitt prestationsresultat med spelarens – i övrigt agerar den helt på egen basis. De två element som beskrivs i följande stycke fungerar dock fortfarande som en mall för problemmetoden i det här projektet.

Abraham och McGee (2010) nämner två huvudsakliga element i ett anpassningsbart medhjälparsystem som översatt till svenska är *en lagkamrats- och motståndarvärderare* samt *en justeringsmekanism av lagkamrater*. Syftet med det förstnämnda är att mäta spelarens och motståndarens prestation (från medhjälpar-AI:ns perspektiv) medan det sistnämnda ska stötta spelaren (indirekt) och i och med det främja lagframgång. Enligt Aswin och McGee (2010) förses spelaren med en spelupplevelse av både utmaning och stödjande, till följd av den samevolution som sker mellan de olika parterna.

De två beskrivna elementen tillämpas i det här projektet (se metodbeskrivningen i kapitel 3.2) och detta i kombination med det andra tillvägagångssättet som beskrivs i kapitel 2.5.3 (*Metod 1: Realtidsevolution med användning av spelspecifik information*), skapar en stabil utgångspunkt för att uppnå ändamålet med det här projektet.

3 Problemformulering

Syftet med projektet är att utveckla och utvärdera en dynamiskt anpassningsbar medhjälpar-AI med hjälp av en genetisk algoritm. AI:n ska alltså anpassa sig efter en spelare i realtid. Spelet är av enklare karaktär men samtidigt tillräcklig för att uppnå sitt syfte som experimentmiljö.

AI:n består av en genetisk algoritm som enligt Negnevitsky (2004) kan definieras som en stokastisk sökalgoritm baserad på den biologiska evolutionen. Det som gör genetiska algoritmer intressanta för uppgiften är att de enligt Bourg och Seemann (2004) underlättar balanseringen av ett spel. Det kan vara en krävande uppgift eftersom skickligheten mellan spelare kan skilja sig väldigt och enligt Tan, Tan och Tay (2011) är det ur utvecklarsynpunkt viktigt att designa en spel-AI som kan satisfiera den variation av spelare som kommer att interagera med spelet.

Bourg och Seemann (2004) nämner även att balansering utgör en stor del inom spelutveckling, vilket motiverar anledningen att undersöka ifall genetiska algoritmer kan utnyttjas för att balansera en medhjälpares AI; balansera i den bemärkelsen att medhjälparen i realtid förhåller sina handlingar till spelaren. Doherty och O'Riordan (2008) promotar också evolutionära tekniker (som är det området genetiska algoritmer tillhör) då det enligt dem är väldigt svårt för spelutvecklare att inte bara hårdkoda taktiker för samarbetande AI-agenter, utan också att avgöra när och var det är effektivt att tillämpa vissa taktiker.

En medhjälpare som inte gör någon större nytta eller kanske till och med förstör för spelaren, upplevs troligtvis som både meningslös och frustrerande. Motsatsen, en för intelligent AI, kan istället försämra spelupplevelsen genom att spelaren får mindre utmaning och ur designsynpunkt är det viktigt att spelarens *flow* bibehålls. Begreppet flow introducerades av Mihály Csíkszentmihályi och är resultatet av hans mångåriga forskning kring olika sinnestillstånd. Csíkszentmihályi (1997) visade till exempel att när någon hamnar i ett flowtillstånd känner sig denne bland annat glad och upprymd. Vidare skriver Csíkszentmihályi (1997) att detta tillstånd infinner sig när förutsättningar att klara av en utmanande uppgift är goda. Med andra ord är det mycket betydelsefullt att medhjälpar-AI:n varken uppfattas som störande eller utmaningshämmande om spelaren ska få möjlighet att bibehålla ett tillstånd i flow. Schell (2008) nämner också flow och menar att det är exakt den känslan en spelskapare vill att spelarna ska få åtnjuta. En bidragande lösning på det problemet kan vara en flexibel AI hos medhjälparen: AI:n ska anpassa sig till spelarens skicklighetsnivå på ett sätt så att den inte utgör ett hinder genom att prestera för dåligt. Medhjälparen ska inte heller prestera bättre än spelaren.

Ett annat problem, ur designsynpunkt, kan vara att ju mer spelaren underpresterar desto sämre blir medhjälparna, vilket till exempel kan bidra till förminskat flow. Så är delvis fallet i spelet i det här projektet eftersom det tar längre tid för en sämre spelare att klara spelet (spelets svårighetsgrad förändras emellertid inte). Det här är dock inget större problem eftersom det främst handlar om att balansera motståndet, snarare än medhjälparna. Genom att dela in ett spel i svårighetsgrader och anpassa motståndet därefter kan medhjälparnas AI förbli densamma, utan att effekten av dem blir att en sämre spelare får det svårare medan en bättre spelare får det lättare.

Det här projektet bidrar till utvecklingen av speldesign genom att experimentera med nya sätt att balansera medhjälpar-AI. Genom utvärdering av den ger arbetet också insikt i *om* och i så fall *hur* genetiska algoritmer kan utgöra en bra grundstomme till en AI i spelsammanhang. Att använda evolution till att balansera en medhjälpar-AI är ett relativt outforskat område och därför ger det här experimentet nya infallsvinklar inom utvecklingen av medhjälpar-AI. Vidare kan evolution med fördel kombineras med andra AI-tekniker (till exempel tillståndsmaskiner eller artificiella neurala nätverk) för att uppnå ytterligare balans.

3.1 Delmål

Projektet är uppdelat i två delmål. Det första uppfyller all implementation medan det andra och sista är utvärdering av den genetiska algoritmen.

3.1.1 Delmål 1: Implementation

Det första delmålet är att skapa miljön med en styrbar karaktär och tre agenter. Spelarkaraktären ska styras av en AI. Alla agenter ska vara baserade på en genetisk algoritm och genom den anpassa sina handlingar efter spelaren. Anpassningen ska ske genom att jämföra relevant data, vilket i det här projektet är poäng, mellan spelaren och agenterna.

Två fördefinierade medhjälpare, eller närmare bestämt genom/kromosomer, ska också implementeras och användas som targets (se kapitel 2.5.3) för de aktiva medhjälparna under en spelomgång.

3.1.2 Delmål 2: Utvärdering

I det andra delmålet ska resultatet utvärderas genom poängstatistik från ett större antal körningar. Poängstatistik innebär här statistik över spelarens och medhjälparnas poäng, där medhjälparnas poängsnitt jämförs med spelarens i ett visst tidssteg under de olika testkörningarna. Poängutvecklingen kommer att ge en tydlig bild av medhjälparnas anpassningsförmåga. Körningarna ska även delas in i tre delar, i vilka spelarens beteende skiljer sig intelligensmässigt. Detta för att kunna utvärdera hur väl AI:n anpassar sig till olika skicklighetsnivåer på spelaren.

3.2 Metodbeskrivning

I det här avsnittet beskrivs metoderna som tillämpas för att uppfylla delmålen (kapitel 3.1) för implementation och utvärdering.

För att underlätta utvärderingen av en AI med stokastiskt beteende, ska den först implementeras eftersom det annars är mycket svårt att veta hur den kommer att bete sig. Anledningen till att tre agenter ska agera i varje spelomgång är för att effektivisera testningen och öka insamlingen av data från en session, utan att orsaka kaos på spelplanen (vilket det möjligtvis hade varit risk för ifall fler än tre agenter skulle agera på planen samtidigt).

För att kunna utvärdera AI:n genom poängstatistik testas den i en egenutvecklad spelmiljö. Målet med spelet är att samla poäng genom att ta upp guldklimpar och undvika minexplosioner. När minor exploderar orsakar de en förlust av spelarens/medhjälparens poäng ifall denne är inom explosionsradien. Ju längre karaktären befinner sig i explosionen desto mer poäng förgås. Spelaren och medhjälparna delar poängpott och spelet är slut när den potten har uppnått en viss summa. Denna spelmiljö täcker ett antal vitala delar som är vanliga i väldigt många spel (idag): slump, motstånd, ett tydligt mål och något som kan liknas

vid till exempel hälsa. Det sistnämnda syftar på poängen som likt hälsa vanligtvis både kan öka och minska.

Poängstatistiken visas i tabellform där snittet av medhjälparnas poäng jämförs med spelarens snittpoäng i ett visst tidssteg under testkörningarna. Eftersom tre olika tester ska göras, där spelarens skicklighetsnivå skiljer sig, blir det totalt tre diagram som visar resultaten från körningarna. De ger en tydlig bild av hur väl medhjälparna har anpassat sig efter spelarens poäng.

De ideala medhjälparna evolveras fram offline tills ett önskat beteende uppnåtts, sedan hårdkodas deras kromosomer för användning till medhjälparnas evolution. Det intelligenta idealet ska kunna undvika i princip alla minorna och samtidigt samla på sig poäng i relativt hög takt. Det motsatta idealet ska samla på sig poäng i relativt låg takt och kunna undvika större delen av minorna.

Som beskrivet i kapitel 2.1 finns det olika tekniker att tillgå för att implementera en AI. Det här arbetet fokuserar dock på genetiska algoritmer eftersom Bourg och Seemann (2004) påpekar att den tekniken är en bra utgångspunkt för att uppnå ett balanserat beteende hos AI:n. Som nämndes i samband med den relaterade forskningen i kapitel 2.5 bygger AI-upplägget på en metod (se kapitel 2.5.3) som beskrivs i Demasis och Cruz (2002) artikel. Sammanfattningsvis använder metoden en genetisk algoritm där fördefinierade AI-agenter är inblandade i evolutionsprocessen. Dessa agenter fungerar i det här projektet som ett intelligent och ett mindre intelligent ideal för medhjälpar-AI:n att sträva efter. Vilket ideal medhjälparen ska eftersträva växlar under en spelsession beroende på spelarens prestation. En spelare med ojämn skicklighetsnivå orsakar till exempel att medhjälparen genomgår fler växlingar jämfört med en spelare med jämn skicklighetsnivå. Här tillämpas det första elementet som beskrivs i Abrahams och McGees (2010) artikel (se kapitel 2.6), då medhjälparen mäter spelarens prestation gentemot sin egen genom att jämföra poängen. Det andra elementet tillämpas i och med evolutionsprocessen som fungerar som en justeringsmekanism av medhjälparna.

För att kunna avgöra om målet med AI:n har uppnåtts granskas resultatet. Om upplevelsen hade varit av prioritet i det här arbetet, hade intervjuer med testpersoner varit en god källa till information eftersom varje spelares upplevelse är individuell.

En annan metod för att utvärdera är att experimentera med olika skicklighetsnivåer på spelaren och föra poängstatistik mellan ett antal körningar för varje skicklighetsnivå. Eftersom det intressanta är att se hur väl AI:n anpassar sig efter spelaren är det endast relevant att jämföra poängstatistik för en viss skicklighetsnivå (poängskillnaden mellan olika skicklighetsnivåer är alltså inte intressant i det här fallet). Resultatet skulle ge en bild av hur väl genetiska algoritmer kan användas för att skapa en flexibel medhjälpare i spelsammanhang.

Gällande prestanda så kan ett mått på algoritmens tidseffektivitet vara en givande utvärderingsmetod, då det kan vara av betydande grad i spel att en AI inte kräver för tunga beräkningar. Enligt Demasi och Cruz (2002) kräver dock inte metoden som AI-upplägget baseras på någon större åtgång på processorkraft, varpå den utvärderingsmetoden inte är aktuell.

Eftersom medhjälpar-AI:ns anpassningsförmåga prioriteras i det här arbetet är poängstatistik mer betydelsefullt än intervjuer, därför faller valet av metod på poängstatistik. Det hade varit intressant att även utvärdera upplevelsen med hjälp av intervjuer men då upplevelsen inte är slutmålet med projektet, till skillnad från god anpassningsförmåga, faller det bort – det skulle bli alltför tidskrävande att genomföra båda utvärderingsmetoderna på ett kvalitativt sätt.

4 Implementation

I det här kapitlet följer en genomgång av realiseringen av metoden för implementation som tillämpas för att uppfylla det första delmålet (se kapitel 3.1.1).

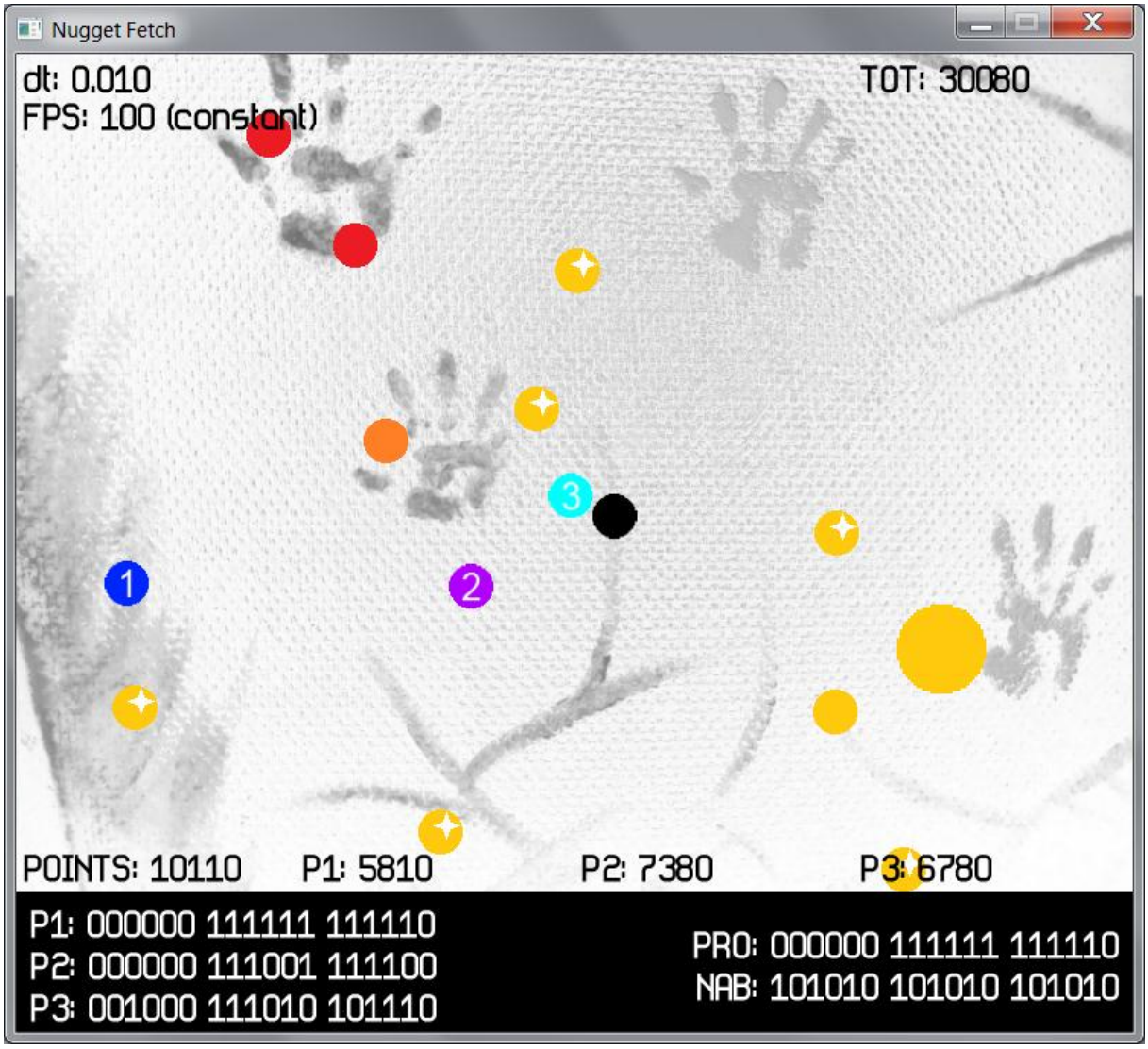
4.1 Experimentmiljön

Den egenutvecklade experimentmiljön är kodad i programmeringsspråket C++ och grafikbiblioteket som används är en del av spelmotorn Haaf's Game Engine (Relish Games, 2008).

Som tidigare nämnts i metodbeskrivningen (kapitel 3.2) går spelet ut på att samla guldklimpar (värda 100 poäng vardera) samtidigt som minexplosioner ska undvikas. Innan en mina sprängs är den ofarlig men tiden tills den sprängs varierar emellertid något och beror på slumpen. En explosion varar dock alltid under lika lång tid och ju längre en spelare befinner sig i en explosion desto mer poäng förlorar den. Spelet är över när den totala poängen, alltså summan av spelaren och medhjälparnas poäng, är ≥ 20000, 25000 eller 30000 beroende på spelarens skicklighetsnivå. Skillnaden i målpoängen började tillämpas för att minska gapet mellan testkörningstiderna: 30000 poäng tar betydligt längre tid att nå för den sämre spelaren än för den bättre, till exempel.

Vilken position en mina eller guldklimp skapas på avgörs av slumpen och det finns inget som hindrar att något hamnar på något annat. Med andra ord kan en mina eller guldklimp dela position med en annan mina eller guldklimp. Maximalt kan det finnas sju stycken minor/explosioner respektive klimpar på plan samtidigt. Anledningen till det är för att undvika "kaos" på planen och på så sätt underlätta observationen av de olika agenternas beteenden.

Figur 4 visar en skärmdump från spelet. Själva spelplanen är 800 x 600 pixlar. Spelaren, medhjälparna, guldklimparna och minorna är alla cirkulära med en radie på 16 pixlar. Explosionerna är också cirkulära men har en radie på 32 pixlar. Det mörka området längst ner, nedanför raden med poäng, hör inte till spelplanen och beskrivs istället i kapitel 4.2.2.



Figur 4 Skärmdump från spelet. Spelaren är den svarta cirkeln och medhjälparna är de numrerade cirklarna (1-3). De små gula cirklarna med vitt "blänk" (totalt fem) är guldklimpar. De resterande fyra små cirklarna är minor i olika faser. Den stora gula cirkeln är en explosion i sista fasen.

Under en körning skrivs var tredje sekund poäng- och genomdata för spelaren och agenterna till en textfil. Detta ska användas till utvärderingen av projektet.

4.2 AI:n

Det här delkapitlet går inledningsvis igenom AI:n som styr spelaren respektive medhjälparna. Detta följt av en beskrivning av hur ett genom är uppbyggt samt avläses för att hantera AI:ns beteende. Därefter följer ett delkapitel om balanseringen av AI:ns anpassningsförmåga. Kapitlet avslutas med en genomgång av koddesignen.

4.2.1 Spelaren

Spelaren implementerades först som en styrbar karaktär men som förberedelse inför utvärderingen och för att en seriös balansering med tre olika skicklighetsnivåer på spelaren skulle bli genomförbar, blev det nödvändigt att implementera en AI-styrd spelare. I det här

projektet håller en AI-styrd spelare troligtvis en högre grad av konsekvent beteende än en mänskligt styrd.

Spelaren styrs alltså av en AI som finns i tre olika skicklighetsnivåer; dessa nivåer är svag, standard och stark. Skicklighetsnivån avgörs av genomen, vilka evolverades fram offline tills tre önskade beteenden (en för respektive nivå) uppnåddes. Att evolvera fram genomen offline är egentligen inte nödvändigt i det här projektet eftersom genomen är såpass små men med större genom hade det varit av betydande grad då det blir svårt att förutse vilka genkombinationer som resulterar i vilka beteenden. Som underlag för besluten jämfördes i alla fall genomens förmåga att samla poäng under en viss tid, snarare än att studera de faktiska beteendena. Spelaren har alltså ett förbestämt beteende och detta förändras aldrig under en körning eftersom tanken är att den ska hålla en relativt jämn nivå hela spelomgången igenom. Detta ska spegla en mänsklig spelare som troligtvis skulle hålla en mer eller mindre jämn nivå i spelet, eftersom varken motståndet eller spelplanen genomgår några större förändringar under en spelomgång.

Den svarta cirkeln och siffran efter "POINTS" i Figur 4 representerar spelaren respektive spelarens poäng.

4.2.2 Medhjälparna

Det finns tre medhjälpare i spelet som likt spelaren är representerade med cirklar. De är dock numrerade 1-3 och har färgerna mörkblå, lila respektive turkos (se Figur 4). Varje medhjälpares poäng finns på samma rad som spelarens ("POINTS") där siffran efter "P1" är poängen som medhjälpare nummer 1 (den mörkblå) har, och så vidare.

Till skillnad från spelaren så förändras medhjälparnas beteende under en körning eftersom de ska anpassa sig efter spelaren, eller snarare spelarens poäng. Med andra ord förändras deras genom/kromosom vid behov, dock inte oftare än var tredje sekund då evolveringsfunktionen anropas. Anledningen till att en timer implementerades är för att undvika att medhjälparna evolverar för snabbt samt för att spara in på processorkraft. Det sistnämnda är inte direkt relevant i detta projekts enkla spelmiljö men i större och mer avancerade spel skulle skillnaden bli väldigt märkbar.

Det mörka området längst ner på skärmdumpen (se Figur 4) visar hur medhjälparnas genom ser ut i realtid, vilka som tidigare nämnts är binära strängar. De två genom som redovisas till höger i samma område är de två targets, eller ideal, som används som överkorsningspartners (se kapitel 2.5.3 för en noggrannare genomgång av targets). Precis som spelarens genom är dessa targets förbestämda, statiska och framevolverade offline. "PRO" är det intelligenta av dem två medan "NAB" är det mindre intelligenta.

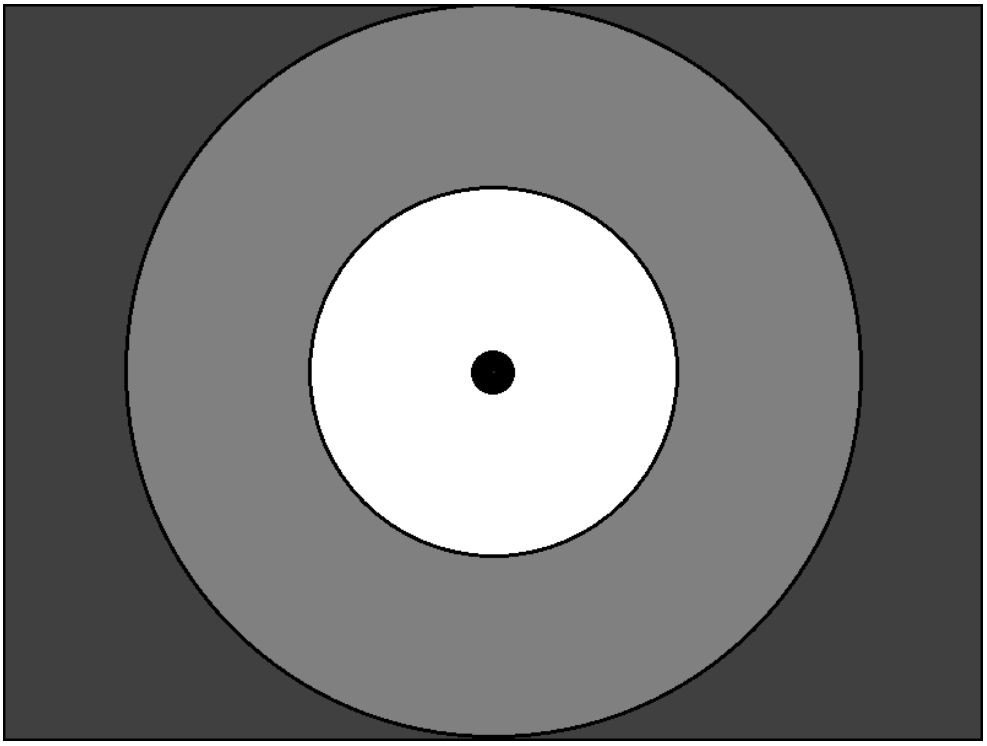
Vid uppstart slumpas medhjälparnas genom fram men genom överkorsning med antingen "PRO" eller "NAB" följt av mutation, växlar medhjälparna intelligensnivå under spelets gång. Vilket ideal medhjälparen korsas med beror på antalet poäng i förhållande till spelarens poäng. Evolveringsfunktionen anropas endast om medhjälparen har mindre än 50% eller mer än 80% av spelarens poäng.

4.2.3 Uppbyggnad och hantering av genom

Spelaren och medhjälparna, det vill säga agenterna, baseras på en genetisk algoritm vilket satisfierar det sista kravet för att uppfylla det första delmålet (se kapitel 3.1.1). Med

agentdesignen i *Online Coevolution for Action Games* (Demasi & Cruz, 2002) som grund, kan AI:n i det här projektet beskrivas med följande:

- Varje agent har två targets att ta hänsyn till: guldklimpar och minor. *Observera att dessa targets skiljer sig från de targets, de så kallade idealen, som används i överkorsningssyfte.*
- Avståndet d (i pixlar) mellan en agent och ett target delas upp i tre avstånd: nära ($0 \leq d < 150$), medellångt ($150 \leq d < 300$) och långt ($d \geq 300$). Se Figur 5 för en överblick.
- En agent kan handla på tre olika sätt: *undvik närmaste mina*, *gå mot närmaste guldklimp* eller *gå till en slumpmässig position*.

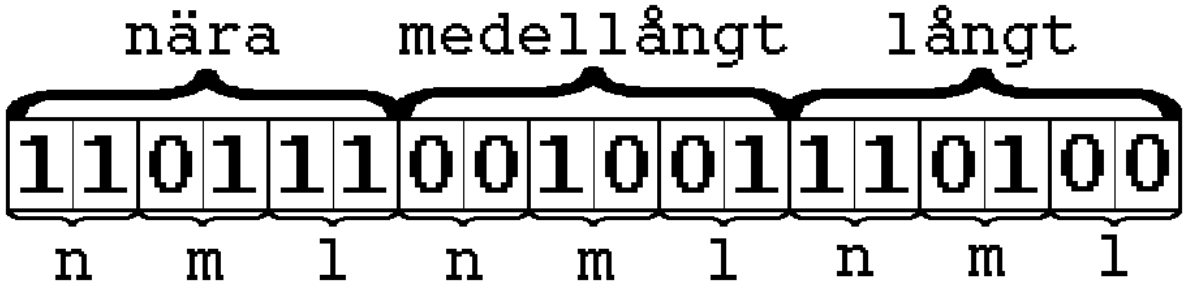


Figur 5 En överblick av avståndszonerna relativa till varandra och spelplanen. Den svarta cirkeln i mitten föreställer en agent. Eventuella minor/guldklimpar inom den vita och ljusgrå zonen befinner sig på nära respektive medellångt avstånd. Minor/Guldklimpar på resterande zoner (de mörkgrå) befinner sig på långt avstånd.

Totalt är det alltså nio (3^2) kombinationer av regler på formen: ***om*** *target*$_1$ *är på avstånd*$_1$ ***och*** *target*$_2$ *är på avstånd*$_2$ ***så utför en handling***. De tre olika handlingarna kodas med hjälp av ett binärt par på formen 00, 01, 10 samt 11, vilket innebär att det finns plats för ytterligare en handling (totalt fyra). På grund av detta kommer två av handlingarna att vara: *gå till en slumpmässig position*. De binära paren avläses av agenten på följande vis:

- 00: *undvik närmaste mina*
- 01 samt 10: *gå till en slumpmässig position*
- 11: *gå mot närmaste guldklimp*

Sammanfattningsvis kommer en agents genom/kromosom att bestå av 18 binära gener: nio olika regler som resulterar i fyra olika handlingar, där en handling beskrivs med ett binärt par. Totalt blir det alltså 18 (9*2) gener. Vidare delas genomet in i tre lika stora delar (sex gener) som i sin tur delas in i tre par. Beroende på avståndet till den mina som ligger närmast hoppar algoritmen till en viss del i genomet: den första delen ifall minan är på nära avstånd, den andra delen ifall minan är på medellångt avstånd och den tredje och sista delen ifall minan är på långt avstånd. Vilket par av binära gener i den utvalda delen som sedan avläses bestäms av avståndet till den guldklimp som ligger närmast agenten. Om till exempel den närmaste minan och guldklimpen båda ligger på nära avstånd ($0 \leq d < 150$) till agenten, ska det *första* paret (av totalt nio) i genomet avläsas. Skulle minan och guldklimpen istället ligga på medellångt ($150 \leq d < 300$) respektive långt ($d \geq 300$) avstånd, avläses det *sjätte* paret. Figur 6 förtydligar vilka villkor som bestämmer vilket genpar som ska avläsas:



Figur 6 Avläsning av genom. De tre övre klamrarna visar vilken del av genomet algoritmen hoppar till först, vilket beror helt på avståndet till den närmaste minan. Därefter väljs ett genpar ut (paren är markerade med små klamrar nedanför genomet). Vilken som väljs beror på avståndet till den närmaste guldklimpen ("n", "m" och "l" står för "nära", "medellångt" respektive "långt").

Exempel: En agent har genomet 110111 001001 110100. Den närmaste minan och guldklimpen ligger på långt respektive nära avstånd. Eftersom minan är på långt avstånd hoppar algoritmen först till del tre i genomet (110100), därefter hoppar algoritmen vidare till par nummer två i den delen då guldklimpen ligger på medellångt avstånd. Par nummer två i del tre är 01 och därmed är agentens nästa handling att *gå till en slumpmässig position*.

4.3 Balansering

För att spelet ska vara en fungerande experimentmiljö är balansering av beteendehanteringen och allt evolutionsrelaterat nödvändigt, då detta har stor inverkan på AI:ns förmåga att anpassa sig.

Den vitala delen i beteendehanteringen är värdena på gränserna som skiljer de olika avståndsvariablerna åt, det vill säga vilket avståndsintervall mellan agenten och närmaste mina/guldklimp som räknas som *nära*, *medellångt* respektive *långt*. Som nämnt i kapitel 4.2.3 är dessa gränsvärden 150 och 300, vilka är samma värden som Demasi och Cruz (2002) använder i sin experimentmiljö. Deras val av gränsvärden fungerar som en bra utgångspunkt även i det här arbetets experimentmiljö, då båda miljöernas spelplaner består av ett enda rum och där de relativa storlekarna på karaktärerna gentemot spelplanets storlek påminner om varandra. Andra gränsvärden testades men skillnaden blev inte direkt märkbar, förutom då värdena var väldigt omotiverade sett till spelplanets storlek: genomuppbyggnaden

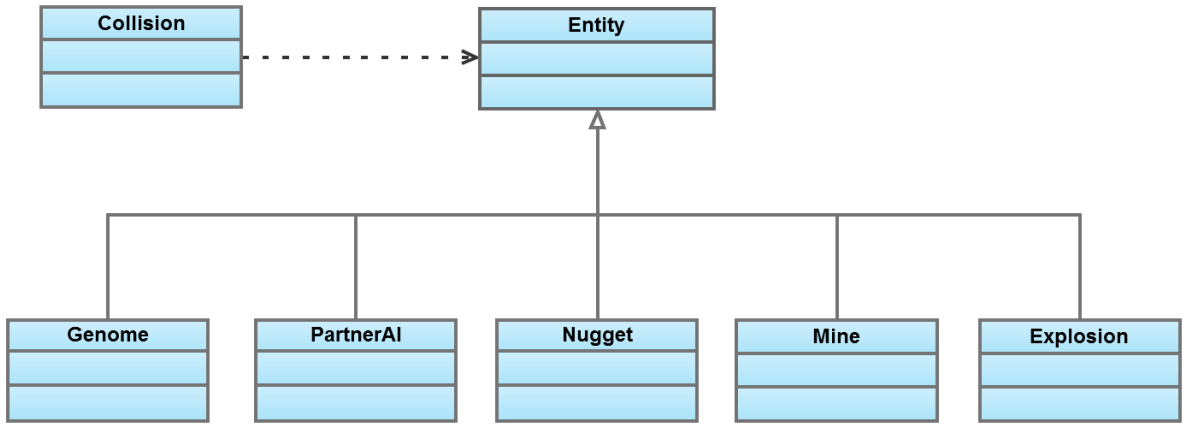
förutsätter att det finns tre olika avståndsintervaller och skulle ett intervall då göras mycket mindre än övriga, skulle de relaterade generna inte utgöra någon större funktion.

Evolveringsfunktionen anropas av den aktuella medhjälparen ifall dess poäng är lägre än 55% eller högre än 75% av spelarens, det vill säga ett intervall på tio procentenheter mindre än det önskade poängintervallet på 50-80% (se kapitel 4.2.2). Anledningen till det är att evolveringen förväntas ta lite tid, vilket motiverar behovet av en marginal.

I evolveringsfunktionen sker både överkorsning och mutation. Sannolikheten att en mutation inträffar är 5% och då ändras endast en gen på slumpmässig position i genomet. Sannolikheten att en överkorsning sker, den så kallade korsningsfrekvensen, är mycket högre; nämligen 75%. Som nämnt i kapitel 2.2 är överkorsning den huvudsakliga operationen i evolutionsprocessen för genetiska algoritmer, därav den höga sannolikheten. Överkorsningsmetoden som används är multipunktskorsning (se kapitel 2.3.3) med tre punkter. Med andra ord kan allt från ingen gen upp till tre gener (av totalt 18) ändras eftersom ingen hänsyn tas till det faktum att en gen på en viss position kan vara likadan hos den aktuella medhjälparen som hos dess partner ("PRO"/"NAB"). Om medhjälparen har under 55% av spelarens poäng behöver den bli intelligentare, något som medför att "PRO" väljs ut som partner vid anrop av evolveringsfunktionen. Skulle medhjälparen ha över 75% av spelarens poäng väljs istället "NAB" ut som partner då medhjälparen behöver bli mindre intelligent.

4.4 Koddesign

I det här kapitlet presenteras klassdiagram (se Figur 7) för programmet (spelet), följt av en beskrivning av varje klass.



Figur 7 Klassdiagram för programmet.

Nedan följer en kort beskrivning av varje klass:

- Genome. Representerar ett genom/en kromosom med hjälp av en lista med gener, det vill säga en lista med 0:or och/eller 1:or. I konstruktorn för klassen tilldelas listan slumpade värden.
- Entity. Abstrakt basklass för alla entiteter. Spelaren, medhjälparna, guldklimparna, minorna och explosionerna är alla entiteter.

- PartnerAI. Representerar en medhjälpar-AI. Här definieras också evolveringsfunktionen: eftersom genomen på idealen "PRO" och "NAB" är förbestämda finns dessa definierade i *PartnerAI*, vilket medför att varje medhjälpare kan ta hand om sin egen evolvering. Även spelar-AI:n är en instans av denna klass.

- Nugget. Representerar en guldklimp.

- Mine. Representerar en mina.

- Explosion. Representerar en explosion.

- Collision. Kollisionsdetektering mellan två entiteter av cirkulär form.

Main är själva navet i programmet; här skapas spelmiljön och instanser av spelaren och medhjälparna. De sistnämnda i form av en lista med *PartnerAI*. I *Main* anropas även alla uppdaterings- och renderingsfunktioner. Undersökning av kollisioner sker också i *Main*; de kollisioner som undersöks är mellan spelare och minor, spelare och guldklimpar, medhjälpare och minor samt medhjälpare och guldklimpar.

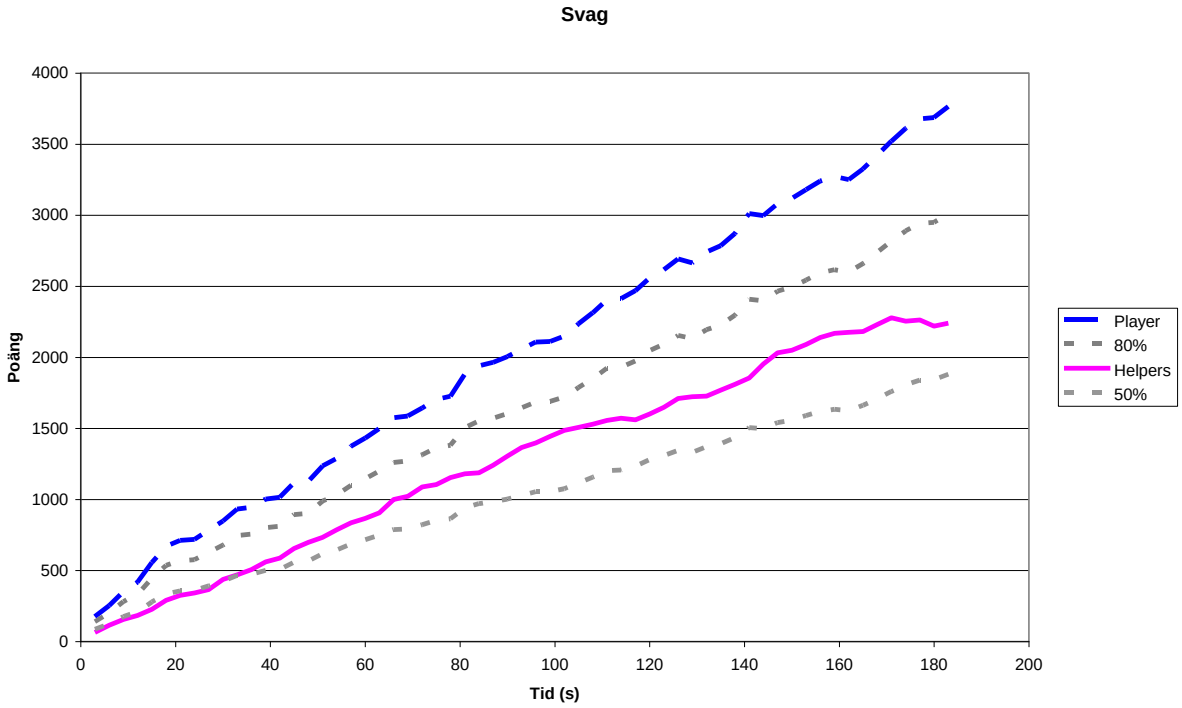
5 Analys

Syftet med projektet är att utvärdera anpassningsbarheten hos medhjälpar-AI:n; resultatet av denna utvärdering redovisas och analyseras i det här kapitlet.

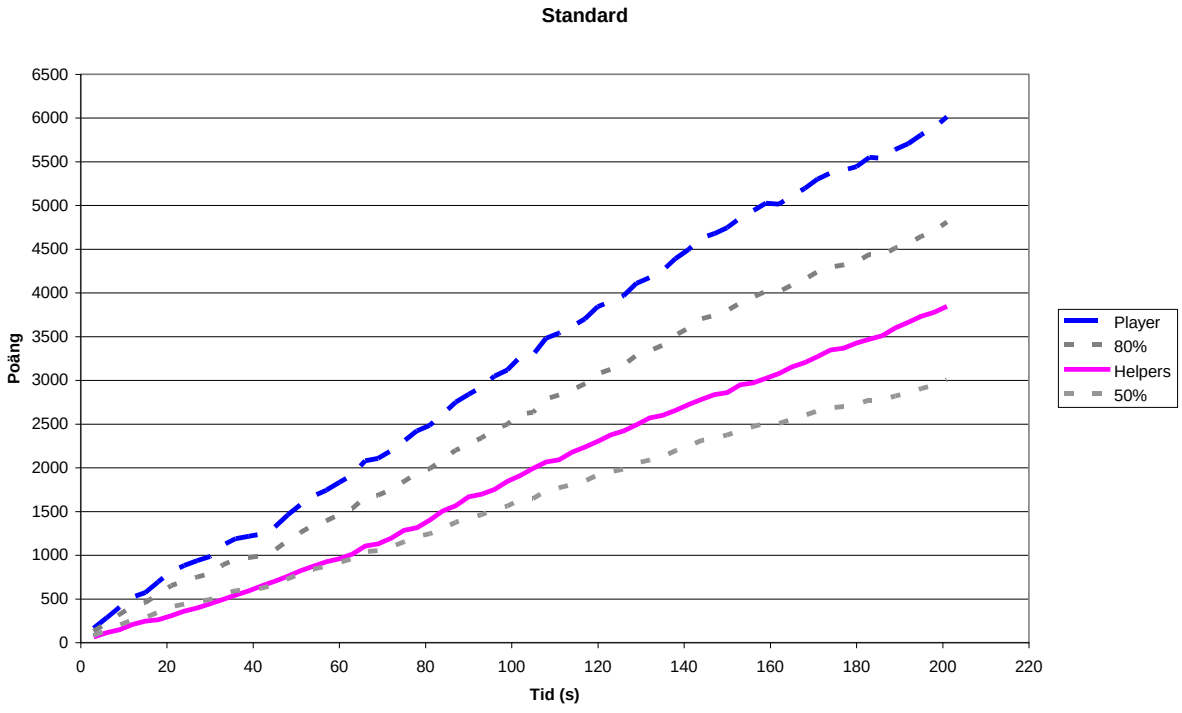
För att uppfylla delmål 2 (kapitel 3.1.2) utvärderas resultatet genom poängstatistik från ett större antal körningar, närmare bestämt 75 (25 x 3). Körningsresultaten är uppdelade i tre delar, där intelligensnivån på spelaren skiljer sig. För att längden på en körning inte ska skilja sig för mycket mellan de olika nivåerna, skiljer sig målpoängen mellan de tre körningsgrupperna: med en svag spelare är den 20000, med en standardspelare är den 25000 och med en stark spelare är målpoängen 30000.

Under en körning skrivs medhjälparnas genomsnittliga poäng samt spelarens poäng till två olika textfiler var tredje sekund. Efter en körning lagras resultatet i ett Excel-dokument. Eftersom körningstiderna beror på hur väl spelaren och medhjälparna presterar kan dessa skilja sig en hel del; för att utvärderingen då ska kunna genomföras korrekt täcker endast resultatdiagrammen det tidsintervall som alla körningar uppfyller. Med andra ord: om två körningar har körningstiderna 180 respektive 200 sekunder används endast värdena till och med 180 sekunder in i körningarna. De värdena från tidsintervallet 181-200 sekunder i den sistnämnda körningen redovisas alltså inte i resultatdiagrammet. Ett alternativ hade varit att implementera en timer (till exempel 5 minuter) i spelet, så att alla körningar fått en fast tid. Det hade egentligen varit bättre eftersom inga resultatvärden då gått till spillo.

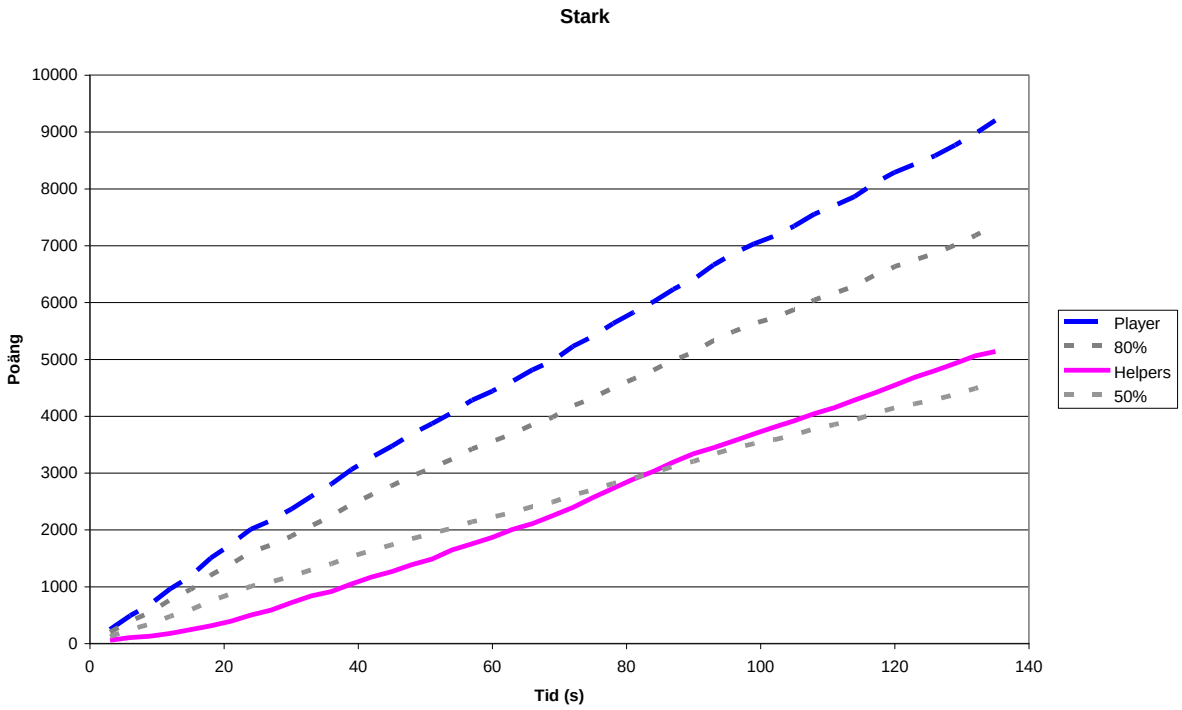
Figur 8, Figur 9 och Figur 10 visar resultatdiagrammen från de tre olika gruppkörningar som genomfördes med en svag spelare, en standardspelare respektive en stark spelare. Observera att inga balanseringsvärden har ändrats mellan de olika körningarna, då experimentet är tänkt att visa hur väl medhjälpar-AI med lika förutsättningar kan anpassa sig till olika nivåer hos en spelare. Den streckade översta linjen i varje diagram visar spelarens poängkurva, alltså spelarens genomsnittliga poängutveckling under ett visst tidsintervall in i de 25 körningarna. Den andra helstreckade linjen visar medhjälparnas poängutveckling under samma tidsintervall. De streckade linjerna, ovanför samt nedanför medhjälparnas, visar inom vilket område medhjälparnas kurva ska hålla sig inom. Det området representerar med andra ord poäng som är 50-80% av spelarens, vilket är den poängkvot medhjälparna ska sträva efter att ha under varje körning.



Figur 8 Resultatdiagrammet för en spelare på lägsta skicklighetsnivån (svag).



Figur 9 Resultatdiagrammet för en spelare på mellersta skicklighetsnivån (standard).



Figur 10 Resultatdiagrammet för en spelare på högsta skicklighetsnivå (stark).

Figur 8 och Figur 9 visar att medhjälparna håller en sund poängutveckling som endast ligger strax under 50%-gränsen de första 30 respektive 40 sekunderna. Det sistnämnda beror troligtvis på att slumpen avgör medhjälparnas startgenom, vilket medför att dessa kan vara väldigt missanpassade till uppgiften i fråga – tills genomen evolverats ett antal gånger. Att genomen aldrig ligger i överkant (vid 80%) är antagligen av samma anledning. Sannolikheten att slumpen orsakar dåliga gener vid start är större än tvärtom och när en medhjälpare som haft dåliga startgener väl uppnår villkoret att ha en poäng som är ≥ 55% (se kapitel 4.3) av spelarens, kommer evolveringen att stanna upp tills något villkor bryts. Om villkoret uppnås av ett genom som är precis tillräckligt bra för att hålla medhjälparens poäng ≥ 55% av spelarens betyder det att medhjälparen inte kommer att vara mer än duglig – tills villkoret bryts och en ny evolvering genomförs, alternativt en betydande mutation inträffar.

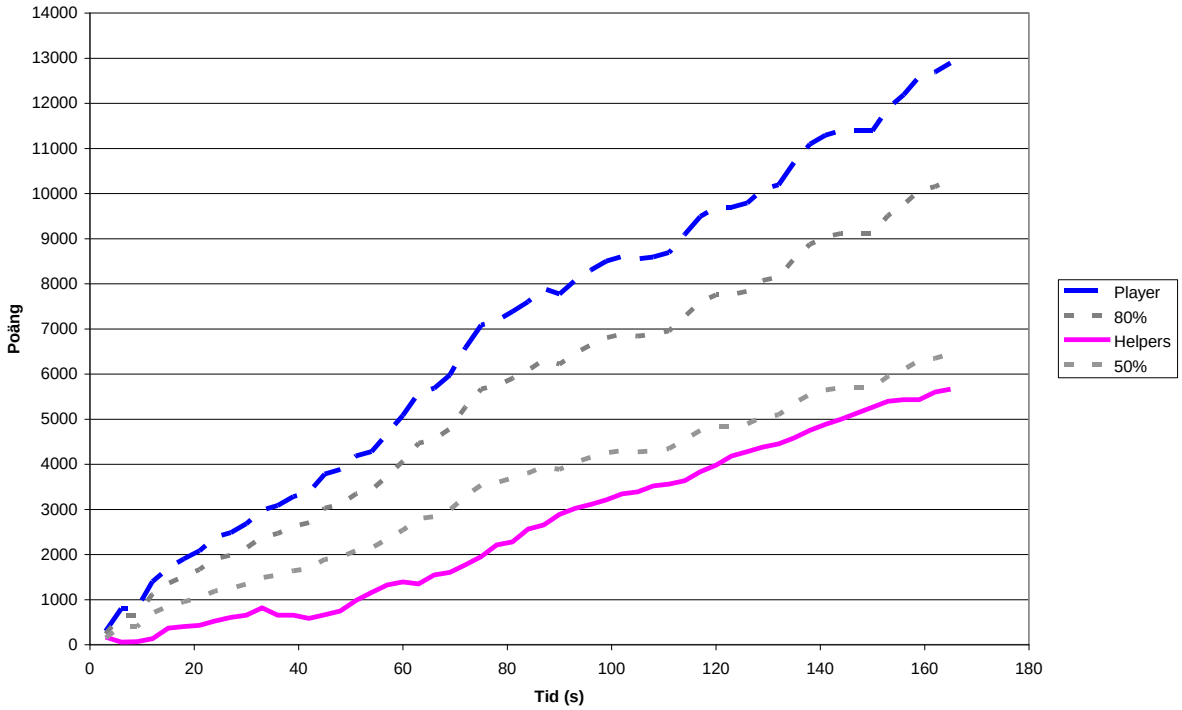
Figur 10 stärker teorin ytterligare eftersom medhjälparnas poängkurva ligger en bra bit under 50%-gränsen de första 70 sekunderna: ju bättre spelaren är desto tydligare blir skillnaden mellan denne och medhjälparna i början av en körning, då den starka spelaren saknar den slumpmässiga handling som orsakas av genkombinationerna 01 och 10. Med andra ord har den starka spelaren, till skillnad från de två andra, alltid ett tydligt mål – oavsett situation – vilket innebär att den jagar guldklimpar i princip hela tiden och undviker endast minor ifall de utgör ett riktigt hot. Den starka spelaren är alltså stabil redan från start vilket medför att medhjälparna ofta får en väldigt trög start och det tar tid för dem att komma ikapp.

Utöver problemet med startgenomen visar resultaten att det med användning av genetiska algoritmer finns god potential att åstadkomma fungerande dynamisk anpassningsbarhet av medhjälpar-AI i spel. För att ytterligare motivera potentialen, är en idé att låta guldklimparna istället representera fiender som besegras genom att medhjälparna eller

spelaren åker på dem. Fiender som ska besegras är ett klassiskt element i spel och att kunna relatera experimentmiljön till det medför att resultaten blir än mer intressanta.

Problemet med startgenomen är också relativt enkelt att eliminera genom att använda förbestämda startgenom som evolverats fram offline; på så sätt undviks de riktigt dåliga genomen. Det här blir väsentligare ju större genomen är eftersom det då tar längre tid att justera dem med hjälp av evolution.

Att använda genomsnitt i slutresultaten är ett bra sätt att redovisa medhjälpar-AI:ns potential men faktum är att det kan dölja enstaka resultat som är riktigt dåliga; Figur 11 visar ett exempel på detta.



Figur 11 Det sämsta resultatet av de 75 körningarna.

I Figur 11 visar resultatet att medhjälparna misslyckats att anpassa sig hela körningen igenom. Att en stokastisk algoritm såsom den genetiska algoritm medhjälpar-AI:n bygger på i det här projektet "misslyckas", som Figur 11 visar, är något som måste tas med i beräkningarna. Det är också något som givetvis vill undvikas hos AI i spel och i kommande diskussion (kapitel 6.2) ges förslag på hur det här problemet skulle kunna tacklas.

Balanseringen av AI:n kan till en början uppfattas som ytterligare ett stort problem men faktum är att mycket av den kan automatiseras genom att evolvera fram genomen offline, vilket är en stark fördel med genetiska algoritmer. Dock gäller det att korsnings- och mutationsfrekvenserna hålls på en förnuftig nivå och då är det en fördel att ha följande i åtanke: ett bra utgångsläge är att hålla mutationsfrekvensen låg (se kapitel 2.3.4) och korsningsfrekvensen hög, då den sistnämnda är den huvudsakliga operationen i genetiska algoritmer (Hoffmann, 1998). I det här projektet har framför allt avståndsgränserna (se kapitel 4.2.3 samt 4.3) haft stor inverkan men dessa behöver inte vara svårbalanserade; det är mycket möjligt att evolvera fram passande genom offline så länge utgångsvärdena för avståndsgränserna är realistiska (sett till spelplanen).

6 Slutsatser

Kapitel 4: *Implementation* och kapitel 5: *Analys* visar att projektets alla delmål har uppnåtts. Ett spel i 2D med en AI-styrd spelare och tre AI-styrda medhjälpare har skapats; alla baserade på en genetisk algoritm. En medhjälpare anpassar sig också dynamiskt efter spelaren genom att regelbundet jämföra poängen dem emellan och evolvera vid behov. Två fördefinierade genom, ett intelligent och ett mindre intelligent, har också implementeras och används som partners till medhjälparna under evolveringen, då deras gener mixas. I och med detta uppfylls det första delmålet.

Utvärderingen har genomförts med hjälp av poängstatistik från ett större antal körningar, närmare bestämt 75 stycken: 25 med en stark spelare, 25 med en standardspelare och 25 med en svag spelare. Poängstatistiken är över spelarens och medhjälparnas poäng, där medhjälparnas poängsnitt jämförs med spelarens i ett visst tidssteg under de olika testkörningarna. Eftersom körningar är indelade i tre grupper (beroende på spelarens intelligensnivå) ger poängutvecklingen en tydlig bild av medhjälparnas anpassningsförmåga till olika skicklighetsnivåer på spelaren, vilket uppfyller det andra och sista delmålet.

6.1 Resultatsammanfattning

Den experimentmiljö som utvecklats och använts under utvärderingen är ett enkelt actionspel där en AI-styrd spelare tillsammans med tre AI-styrda medhjälpare ska samla guldklimpar som ger poäng. Det finns även minor på planen och när de exploderar förlorar spelaren och medhjälparna poäng ifall de befinner sig inom explosionsradien.

Spelaren och medhjälparna baseras på en genetisk algoritm och med hjälp av experimentmiljön har medhjälparnas förmåga att anpassa sig dynamiskt efter spelaren utvärderats. Målet är att medhjälparna ska bibehålla en poäng på 50-80% av spelarens, då tanken är att de inte ska vara för bra eller för dåliga i förhållande till spelaren. För att uppnå detta justerar en medhjälpare sitt beteende under spelets gång genom att evolvera i realtid.

Utvärderingen visar att genetiska algoritmer skulle kunna utgöra en bra grundstomme för AI i spel, vilket var förhoppningen. Medhjälpar-AI:n håller alltså större delen av tiden en önskad poäng på mellan 50% och 80% av spelarens under en körning. Undantaget är när spelarens skicklighetsnivå är väldigt hög, då det tar tid för medhjälparna att komma ikapp. Troligtvis beror detta främst på att startgenomen slumpas fram, vilket är väldigt vanligt när det gäller genetiska algoritmer och evolutionära tekniker i allmänhet. Slumpfaktorn när det gäller överkorsning och mutation påverkar också. På grund av oberäkneligheten som slumpfaktorn medför, kan AI:n ibland få ett oönskat beteende – något som ofta är mycket svårt att förutse. Denna brist gör det nödvändigt att fundera på lösningar som kan komplettera genetiska algoritmer vid tillämpning av dem i spelsammanhang.

I övrigt är resultaten generellt sett väldigt lovande och visar att genetiska algoritmer mycket väl skulle kunna användas för att åstadkomma välfungerande medhjälpar-AI i spel som anpassar sig i realtid.

6.2 Diskussion

Även om utvärderingen visar på god potential är det viktigt att kunna applicera den på dagens spel; att låta guldklimparna i experimentmiljön istället representera fiender gör den

genast mer aktuell. Då går det att relatera till spel som *Gears of War 3* (Epic Games, 2011), *Mass Effect 2* (BioWare, 2010) och *Ratchet & Clank: All 4 One* (2011) av Insomniac Games eftersom alla spelen har fiender som ska besegras med hjälp av medhjälpare. Det är även relativt enkelt att föreställa sig experimentmiljön i 3D, vilket är viktigt då många av spelen idag är i just 3D.

Ur designsynpunkt kan det diskuteras ifall medhjälparna verkligen bör anpassa sig efter spelaren på ett sådant sätt att de presterar lite sämre än denne. Det skulle medföra att sämre spelare får sämre medhjälpare och således blir spelet svårare ju sämre spelaren är. En lösning är att även motståndet anpassas efter spelaren men spelet skulle också kunna delas in i olika svårighetsgrader: en spelare som väljer att köra på en lätt svårighetsgrad skulle då kunna få bättre medhjälpare än den som väljer att köra på en svårare svårighetsgrad.

För att kunna komma ännu närmare dagens spel behöver genomen också innehålla fler egenskaper, till exempel gener för hälsa, styrka och snabbhet som är några av de vanligaste attributen hos en karaktär. Med sådana gener kan det dock bli aktuellt att låta genomsträngarna bestå av heltal (*int*) istället för endast 1:or och 0:or. På så sätt blir genomen mer högnivå, där en siffra direkt kan översättas till exempelvis medhjälparens hälsa. Genomen skulle också kunna bestå av flyttal (*float*); då blir det möjligt att låta en gen representera en procentchans att utföra en viss handling, till exempel att medhjälparen förgiftar fienden vid en eventuell attack. Möjligheterna är många.

Med fler egenskaper hos medhjälparna behövs mer avancerad fitness-evaluering (se kapitel 2.3.1). I det här projektets spelmiljö hade det till exempel kunnat vara hur många fiender medhjälparen besegrar under *x* sekunder, där *x* är en konstant. Detta förutsätter givetvis att medhjälparna har en gen för styrka och att fienderna har någon form av hälsa, annars försvinner poängen med en sådan fitness-evaluering. Ifall medhjälparna haft en gen för hälsa skulle ytterligare en fitness-evaluering kunna vara hur mycket skada en medhjälpare tar under ett visst antal sekunder. Med en sådan definition är alltså spelet svårare ju mer skada medhjälparna tar under en viss tidsperiod.

För att tackla problemet med oberäkneligheten som slumpfaktorn i genetiska algoritmer medför är en idé att kombinera denna icke-deterministiska teknik med en deterministisk, som till exempel tillståndsmaskiner, för att få det bästa av två världar. I det här projektet skulle en sådan kombination kunna innebära att medhjälparna i så stor utsträckning som möjligt går åt ett annat håll än spelaren (se även sammanfattningen av Abrahams och McGees (2010) forskning i kapitel 2.6). Det skulle minska risken för en situation där båda går mot samma guldklimp, då endast en av dem kommer kunna få guldklimpen eftersom det är "först till kvarn" som gäller. Detta skulle troligtvis effektivisera poängsamlandet drastiskt. I andra typer av spel, till exempel rollspel, skulle medhjälparna kunna anta andra roller än spelaren för att komplettera denne *men* att deras beteenden i övrigt styrs av genetiska algoritmer. Då skulle medhjälparen till exempel kunna anta en defensiv roll ifall spelaren är väldigt offensiv (deterministiskt) och sedan låta genetiska algoritmer styra det defensiva beteendet (icke-deterministiskt). På så sätt behålls två viktiga egenskaper som en icke-deterministisk teknik besitter: variation och oförutsägbarhet. Oberäkneligheten blir helt enkelt mer kontrollerad.

En mer avancerad variant skulle kunna vara att implementera (alternativt offline-evolvera fram) ett större antal fördefinierade genomsegment, indelade i olika typer som till exempel *healer* eller *tank*, som sedan kombineras till hela genom med hjälp av en genetisk algoritm.

På så sätt kan medhjälparna anta hybridroller för att ytterligare öka sin anpassningsförmåga; skulle spelaren exempelvis vara svårt skadad skulle medhjälparen kunna få ett av sina genomsegment utbytt till ett av typen healer samtidigt som andra segment bibehålls. Detta skulle också kunna begränsas med regler som förhindrar att en medhjälpare kan anta alla roller vid alla tillfällen; det vore exempelvis inte förnuftigt med en stark tank som samtidigt är en ultimat healer. Med fördefinierade genom underlättas också testningen av medhjälparna eftersom beteendena blir något mer begränsade än om en "traditionell" överkorsning hade genomförts.

Andra fördelar med att tillämpa icke-deterministiska tekniker är att de underlättar för programmeraren eftersom denne inte behöver definiera alla beteenden; istället främjas en viss grad av självutvecklande beteende (Bourg & Seemann, 2004). Det kan därför vara intressant att undersöka hur väl andra icke-deterministiska tekniker, exempelvis ANN, skulle fungera som basis för medhjälpar-AI.

Det är också viktigt att poängtera att genetiska algoritmer kan utgöra en bra grundstomme även för annan typ av AI, som till exempel dynamiskt anpassningsbar motståndar-AI, vilket Demasis och Cruz (2002) experiment visar (se kapitel 2.5 för en sammanfattning).

6.3 Framtida arbete

Framtida arbeten med det här projektet som utgångspunkt är antagligen många och mycket kan relateras till det som står skrivet i föregående kapitel. Att göra medhjälpar-AI:n mer avancerad genom fler attribut, som till exempel hälsa och styrka, är både intressant och väldigt relevant. Det här projektet visar potentialen med genetiska algoritmer när det gäller dynamisk medhjälpar-AI men det behövs ytterligare arbete för att visa hur bra det faktiskt skulle fungera i så avancerade spel som många av dagens spel är. Att utöka medhjälparnas egenskaper skulle vara ett stort steg i rätt riktning.

Att jobba vidare på den genetiska algoritmen och kombinera den med någon deterministisk teknik i samband med att ge medhjälparna fler egenskaper är också väldigt relevant och skulle kunna ge upphov till något form av AI-arkitektur. Det skulle samtidigt skapa en möjlighet att göra något åt problemet med startgenomen som orsakar att medhjälparna i vissa fall har svårt att komma ikapp.

Annat framtida arbete skulle kunna vara att undersöka hur väl medhjälpar-AI:n fungerar för olika typer av spel. För att det ska bli intressant är dock, återigen, fler egenskaper hos en medhjälpare nödvändigt. Medhjälpare i ett rollspel skulle antagligen vara i behov av en *mana*-gen och en medhjälpare i en förstapersonsskjutare behöver troligtvis en gen för mängden ammunition denne har till sitt förfogande.

En undersökning av processorkrafts- och minnesåtgång skulle också vara givande, ifall arbetet skulle utvecklas ytterligare. Eftersom medhjälparna evolverar i realtid blir det väsentligt att se till att prestandakraven inte är för höga, inte minst ifall spelet skulle ha ett flerspelareläge över nätverk. Om till exempel fitness-funktionen behöver köras ofta är det viktigt att den inte är för komplex. Detta leder in på nästa förslag på framtida arbete som är att undersöka olika fitness-funktioner för olika typer av medhjälpar-AI i samband med en mer avancerad version av medhjälpar-AI:n i det här projektet.

Hur medhjälpar-AI:n upplevs av spelaren är också väsentligt att undersöka. Det här arbetet har fokuserat på medhjälparnas anpassningsförmåga i hopp om att väcka idéer som kan förbättra spelupplevelsen ytterligare men det skulle givetvis vara intressant med en utvärdering av en spelares personliga upplevelse som komplement. I slutändan är det ändå den personliga upplevelsen som räknas men för att kunna utreda den är det nödvändigt med ett arbete som visar fungerande exempel på medhjälpar-AI som anpassar sig i realtid, i vilket det här projektet har varit ett första steg.

Referenser

Abraham, A. T. & McGee, K. (2010) AI for dynamic team-mate adaptation in games. *2010 IEEE Symposium on Computational Intelligence and Games (CIG)*. 18 August IEEE, s. 419–426.

BioWare (2010) *Mass Effect 2* (Version: 1.0) [Datorprogram]. Electronic Arts, Inc.

Bourg, D. M. & Seemann, G. (2004) *AI for Game Developers*. 1st edition. O'Reilly Media.

Buckland, M. (2002) *AI Techniques for Game Programming*. 1st edition. Course Technology PTR.

Csíkszentmihályi, M. (1997) *Finna Flow*. 2:a pocketutgåvan. Stockholm, Natur och Kultur.

Demasi, P. & Cruz, A. J. de O. (2002) Online coevolution for action games. *Proceedings of The 3rd International Conference on Intelligent Games And Simulation*. London, s. 113–120.

Doherty, D. & O'Riordan, C. (2008) Evolving team behaviours in environments of varying difficulty. *Special Issue on the 18th Artificial Intelligence and Cognitive Science Conference (AICS 2007): Part I*. Springer, s. 233-244.

Epic Games (2011) *Gears of War 3* (Version: 1.0) [Datorprogram]. Microsoft Game Studios.

Hoffmann, F. (1998) Incremental tuning of fuzzy controllers by means of an evolution strategy. I: J. R. Koza et al. (red:er), *Genetic programming* (s. 843-851). Proceedings of the third annual genetic programming conference, 22-25 juli, 1998, Madison, Wisconsin.

Insomniac Games (2011) *Ratchet & Clank: All 4 One* (Version: 1.0) [Datorprogram]. Sony Computer Entertainment, Inc.

Irrational Games (2007) *Bioshock* (Version: 1.0) [Datorprogram]. 2K Games.

Lionhead Studios (2001) *Black & White* (Version: 1.0) [Datorprogram]. Electronic Arts, Inc.

Marvelous Interactive (2005) *Harvest Moon DS* (Version: 1.0) [Datorprogram]. Natsume, Inc.

Mitchell, M. (1998) *An Introduction to Genetic Algorithms*. Third Printing. A Bradford Book.

Namco (1980) *Pac-Man* (Version: 1.0) [Datorprogram]. Namco, Inc. & Midway Games, Inc.

Nationalencyklopedin (2012) Tillgänglig på Internet: http://www.ne.se/ [Hämtad Februari 17, 2012].

Negnevitsky, M. (2004) *Artificial Intelligence: A Guide to Intelligent Systems*. 2nd edition. Addison Wesley.

Nintendo EAD (1998) *The Legend of Zelda: Ocarina of Time* (Version: 1.0) [Datorprogram]. Nintendo, Inc.

Relish Games (2008), *Haaf*'s *Game Engine* (Version: 1.81) [Datorprogram]. Tillgänglig på Internet: http://hge.relishgames.com/ [Hämtad April 4, 2012].

Rockstar North (2008) *Grand Theft Auto IV* (Version: 1.0) [Datorprogram]. Rockstar Games, Inc.

Schell, J. (2008) *The Art of Game Design: A book of lenses*. 1st edition. Morgan Kaufmann.

Tan, C. H., Tan, K. C. & Tay, A. (2011) Dynamic Game Difficulty Scaling Using Adaptive Behavior-Based AI. *IEEE Transactions on Computational Intelligence and AI in Games*. 3 (4), s. 289–301.