Reinforcement Learning AI till Fightingspel

Richard Borgstrand

Patrik Servin

Sammanfattning

Utförandet av projektet har varit att implementera två stycken fightingspels Artificiell Intelligens (kommer att förkortas AI). En oadaptiv och mer deterministisk AI och en adaptiv dynamisk AI som använder reinforcement learning. Detta har utförts med att skripta beteendet av AI:n i en gratis 2D fightingsspels motor som heter ”MUGEN”. AI:n använder sig utav skriptade sekvenser som utförs med MUGEN’s egna trigger och state system. Detta system kollar om de skriptade specifierade kraven är uppfyllda för AI:n att ska ”trigga”, utföra den bestämda handlingen. Den mer statiska AI:n har blivit uppbyggd med egen skapade sekvenser och regler som utförs delvis situationsmässigt och delvis slumpmässigt

För att försöka uppnå en reinforcement learning AI så har sekvenserna tilldelats en variabel som procentuellt ökar chansen för utförandet av handlingen när handlingen har givit något positivt och det motsatta minskar när handlingen har orsakat något negativt.

Namn1: Richard Borgstrand

Addres1: 371 44 Karlskrona Kungsmarksvägen 69
E-post1: richardborgstrand@hotmail.com

Namn2: Patrik Servin
Addres2: 371 35 Karlskrona Konstapelsgatan 19
E-post2: patrikservin@hotmail.com

Handledare: Johan Hagelbäck
ISSN-nummer:

Innehållsförteckning

Introduktion

Vad är ett fightingspel?

Ett fightingspel är ett spel där spelaren kontrollerar en karaktär på skärmen som i närstrid ska besegra en motståndare i en eller flera rundor. Båda karaktärerna har en hälsomätare som minskar då man blir träffad. Den som först får ner sin motståndares hälsomätare till noll vinner.

f1. "Street fighter 2". Spelaren till höger har tagit mest skada

Det första tillkännagivna fightingspelet heter ”*Heavyweight Champ*”, och släpptes 1976 som ett av Segas många arcade spel[8]. Men det var inte förens på 80-talet som fightingspelen blev populära. “*Karate Champ*” (1984) introducerade, likt pac-man, “AI patterns” till fightingspel men AI:n var så pass dålig att en andra version släpptes[19]. Då var AI i spel inte alls smarta eftersom spelen hade väldigt simpel spelmekanik och spelade främst med en annan spelare istället för en AI. Fightingspel var de första spelen som började introducerade komplexa AI system[10]. De senaste åren så har AI skymts undan och grafik, ljud och fysik i spelen har haft en mer fokus än själva AI:n[14]. I dagsläget så har även de flesta tillgång till internet vilket har gjort att de flesta fightingspel har funktionalitet för att spela online mot andra spelare. Detta minimerar beroendet av en AI. Dock så behövs det fortfarande en datorstyrd AI när internet inte är tillgängligt.

Förskriptad AI

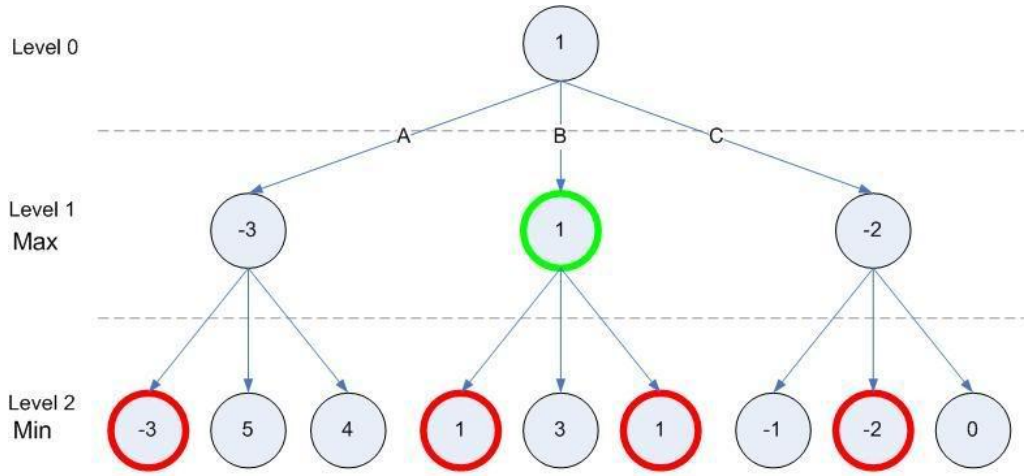
Gamla fightingspel så som Nintendos ”*Kung-Fu*” och Segas *”Mortal Kombat”*, använde sig av förskriptad AI som bestämde vad den skulle göra baserat på vad varje spelare gjorde samt deras position[9]. I dessa mer simpla spel så använde man sig ofta utav en så kallad ”look-up table".
AI:n kunde då i förväg, innan matchen skanna igenom hela listan av attacker och dess respektive frames. Därefter för att sedan kategorisera varje handling t.ex. baserat på attack poäng, defensiva poäng och hastighet i x och y led för att lägga in det i en tabell. När matchen börjar kollar den datorstyrda AI:n vart i förhållande till spelaren den befinner sig och letar sedan upp i tabellen vilken handling som ger mest fördel. Om spelaren t.ex. skulle få AI:n att fastna i ett hörn kan den kolla upp i listan över snabba attacker för att försöka komma ut. När spelaren tillslut lärde sig mönstret som AI:n använde, kunde spelaren besegra AI:n och spelet blev snabbt tråkigt.

Attack	Distance X	Distance Y	Player State
Attack A	10	0	Standing
Attack B	5	-5	Crouching
Attack C	15	10	Air

f2.Ett exempel på ett "look up table"

I de mer avancerade fallen och fightingspelen så utförde AI:n en snabb och kort trädstrukturs sökning för att få reda på den bästa handlingen för den nuvarande situationen. Detta var prestandamässigt väldigt dyrt att göra då sökningen var tvungen att göras under körnings tid. Framtida handlingar är också svåra att beräkna på så kort tid då AI:n inte är stilla utan byter tillstånd (state) ständigt.

Trädstrukturer har använts av AI för att leta reda på vilken väg man skulle ta för att få störst chans att vinna. Till exempel så har Minimax använts för att hitta bästa möjliga sätt att minimera motståndarens maximala handling alltså mest givande handling[13].

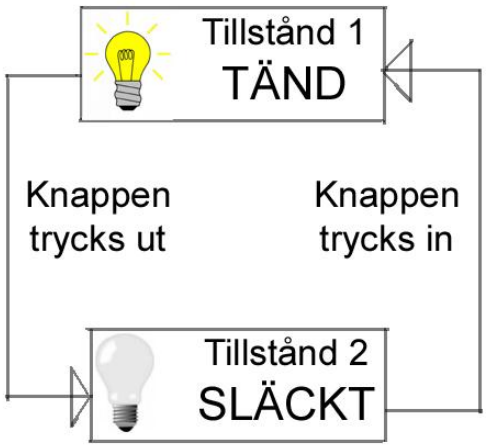


f3. Enligt minimax så är val B bästa valet för att minimera förlusten

Det fungerade bra till spel som schack. Men i dagens fightingspel fungerar det inte lika bra då AI:n kommer till att bli för repetitiv och exploaterbar.

Finite State Machine

Den vanligaste metoden inom fightingspels AI är ”Finite State Machine” (FSM). Den låter AI:n gå från ett tillstånd till ett annat genom att uppfylla vissa krav. Till exempel, vi har en glödlampa med två tillstånd. Den kan antingen vara tänd eller släckt.



f4. Exempel på Finite State Machine

Vi har också ett krav för att glödlampan skall tändas och ett krav för att glödlampan skall släckas. Om en knapp trycks in uppfylls det första kravet och glödlampan tänds. Om knappen trycks ut, uppfylls det andra kravet och glödlampan släcks. Detta kan leda till att AI:n blir förutsägbar, då spelaren ser vilka krav som har uppfyllts, lär sig spelaren viket tillstånd AI:n kommer att hamna i. Det kan vara svårt att skapa en AI som är rolig att spela mot, samtidigt som spelaren inte lär sig hur AI:n spelar. Traditionellt så har AI forskning varit fokuserad på att hitta statiska fixade strategier och bestämda vägar för att maximera chansen till vinst. Detta har gjort att AI med personlighet har hamnat i baksätet för matematiska lösningar[1].

Då attacker i fightingspel varierar i avstånd från låga slag till höga hopp attacker har en stor del varit att försöka förutse vad motståndaren tänker göra likt sten, sax, påse. I nyare spel som ”*Street Fighter*” och ”*Virtua Fighter*” finns det ingen direkt ”rätta” val att göra i en specifik situation[1]. Spelen har spelen blivit mer komplexa och givit fler möjligheter och attacker och därmed sakta gått ifrån sten, sax, påse systemet. Därmed handlar det mer om reaktion och att försöka beräkna vad motståndaren tänker göra.

Detta har gjort att de statiska matematiska algoritmer presterat mycket sämre. Istället så har det öppnat upp mer strategier att försöka läsa spelarens nästa handling eller försöka lura motståndaren att man tänker göra något och göra det motsatta, istället för att kontra “attack a” med “attack b”.

En AI som lär sig är mer oförutsägbar och faller inte in i gropar. Den varierar sig för att vara tillräckligt utmanande och försöker förutse vad motståndaren tänker göra. Dessa problem som hade uppstått kom man fram till att de kunde lösas med en adaptiv AI. Det viktiga är att AI:n varierar sina kombinationer och att det ser flytande ut. Detta gör AI:n roligare att spela mot. Spel som “*Virtua Fighter 4*” och “*Black and White*” har specifikt designats för en adaptiv och lärande AI.

f5.Bild på "Virtua Fighter 4"

Bakgrund

För att lösa dessa problem att AI:n är antingen för lätt eller för svår eller allmänt repetitiv så har man försökt låta AI:n lära sig av spelaren och på så vis få en adaptiv AI. Detta för att försöka åstadkomma ett mer intressantare slutligen roligare spel, vilket även gör att en bättre lärande AI anpassar och försöker förutse vad moståndaren tänker göra. En adaptiv AI behöver inte alltid bli bättre, utan kan anpassa sig efter spelarens erfarenheter. Graepel implementerade reinforcement learning till det kommersiella fightingspelet "*Tao Feng*" till Xbox. Resultatet visade att inlärningstekniker för AI hade god potential för applicering till framtida kommersiella spel[14].

f6. Debug i "Alpha Fighter" med en adaptiv AI

Adaptive AI

Det finns många metoder och formler för en lärande och adaptiv AI:

- Fuzzy logic
- Naive Bayes, Bayesian Learning, Dynamic Bayesian Networks, Hidden Markov Models(HMM)
- Markov Decision Process (Reinforcement Learning)
- Softmax Regression
- N-Gram/sequential prediction
- AI planning (NASA)/Goal-Oriented Action Planning (GOAP) AI, **(***decision-trees*, rule-based *planning*)
- SARSA/Q-Learning - metod/algoritm av reinforcment learning, Temporal Difference learning
- Artificial neural networks (ANN) och genetic algorithms (GA) (långsamma)

Det generella synsättet är att givet ett bestämt antal egenskaper hitta en formel som producerar det optimala resultatet för det nuvarande eller framtida tillståndet baserat på erfarenheten av tidigare tillstånd. Formeln kan vara en linjär kombination av parametrarna för nuvarande tillståndet eller av nuvarande och föregående tillstånd[10]. Mycket av learning baseras på "prediction/pattern recognition" dvs. se mönster som spelaren gör och förutse spelarens nästa handling.

Generell problematik är att AI:n ska snabbt kunna lära sig under kort tid då en match i ett fightingspel i regel bara består av två rundor. Eftersom målet med AI:n är att vara underhållande så får den inte heller vara repetitiv och använda den starkaste attacken om och om igen. Svårighets grad kan alltid balanseras i efterhand. Dessutom så vill man att minne- och processor användning ska vara minimal. I en forskning[1] där ett av målen var att hitta en bra learning algoritm till ett fightingspel så valde man att göra en simpel sten, sax, påse simulation där varje algoritm testades. Testarna valde att välja reinforcement learning pga. av dess snabbhet att lära sig under kort tid. Problem som uppstod då algoritmen applicerades på fightingspels AI:n var att mycket minne krävdes för att spara undan alla tidigare händelser. Slutsatserna var att reinforcement learning fungerade väldigt bra till fightingspels AI. AI:n kunde inte exploateras lika lätt och anpassade sig väldigt snabbt och bra till spelaren. AI:n var dock väldigt svår att möta och besegrade spelaren i de flesta fallen. Detta menade forskarna berodde delvis på att AI:n har perfekt utförande av alla attacker samt att spelet som var testplattformen i sig, inte var av bästa kvalité och oputsat. Andra slutsatser var att många av de learning algoritmer som testades var för prestandakrävande och opraktiska att tillämpa. En tillämpning till reinforcement learning som igenkänning av spelaren och deras spelstil gav tvetydiga resultat om det fungerade bra eller ej.
I en annan studie [11] använde man sig utav spelet "*Alpha Fighter*" och testade en AI modell som var en hybrid av n-grams och Hidden Markov Models som använde sig av fyra olika adaptiva AI modeller på olika sätt. Anledningen till att både använda n-grams och HMM var att n-grams inte var tillräckligt robust algoritm och var antingen för specifik eller för generell. Man uteslöt användande av sequential prediction där resonemanget var att AI:n kunde lätt fastna i en viss attack som fungerade väl. Även om denna attack skulle leda till vinst så kommer AI:n repetitiv använda samma attack om och om igen och inte utforska nya och kanske hitta bättre möjligheter.

I enkät undersökning där tio personer fick spela mot denna hybrid AI och en annan AI utan learning visade det sig att alla aspekter förhöjdes. AI:n var något roligare, verkade mer intelligent men var även lite svårare att möta.

Vid ett test av att använda ett bayesian network i ett eget kodad fightingspel ökade chansen för AI:n att förutse spelarens nästa attack med 30% till skillnad från om den datorstyrda AI:n hade slumpmässigt gissat[15].

*f6. AIBO står för, **A**rtificial **I**ntelligence ro**BO**t*

Alternativt så har adaptiv AI även varit ett sätt att få robotar att lära sig att klara av olika nya förhållanden och även för att låta sig systematiskt testas mot egna svagheter (machine learning).

.

Fördelar och nackdelar med adaptiv AI

- Komplexa skript till en AI har större chans att innehålla design fel. En adaptiv AI kan lösa dessa problem eftersom den lär sig att undvika de felaktiga situationerna[7].
- Bestämda svårighetsgrader på en AI kan ibland inte riktigt representera spelarens färdighet då folk lär sig olika fort. En adaptiv AI kan skalas ner i svårighetsgrad för att dynamiskt balansera och anpassa svårighetsgrad till spelaren under och för kommande matcher[18].
- Minskad produktions kostnad då man slipper ägna tid åt att balansera.
- En adaptiv AI har större potential att bli mer utmanande då den kan lära sig att klara av svåra taktiker.
- Mer intressant och trovärdigt beteende av AI:n, mänskligt liknande.
- Svårt att förutse beteendet av AI:n, mindre kontroll gör det svårare att testa.
- En adaptiv AI behöver nödvändigtvis inte ge bättre resultat.
- Kan ta lång tid att få AI:n att lära sig.
- Mycket minnes åtgång kan krävas för att spara undan all data.

Reinforcement Learning

Reinforcement learning[5] blev populärt på 90-talet inom machine learning och artificiell intelligens men även inom operationsanalys och spreds vidare till psykologi samt neurovetenskap[16]. En Reinforcement AI lär sig av resultatet/konsekvensen av sitt **eget** val, och väljer vidare efter erfarenhet och nya val. Om AI:n gör bra ifrån sig skall han bli belönad med ett antal poäng, och gör den dåligt ifrån sig så får den minus poäng[3]. Det finns olika metoder för Reinforcement Learning, för både långsiktig och kortsiktig konsekvenser. Reinforcement Learning används generellt för att lösa ”Markov decision problem”, där ett val är delvis slumpmässigt och delvis kontrollerat[4]. Ett val leder till en belöning eller en kostnad. Detta för att uppmuntra AI:n till ett beteende att genom “trial and error” välja de mest effektiva handlingarna (som ger mest belöning) givet en situation och utforska nya handlingar för samma eller liknande situationer.

Ett bra exempel på en Reinforcement Learning är spelet *”Black and White”*. Där har spelaren ett djur som spelaren själv får ge beröm eller bestraffa, beroende vad spelaren vill att djuret skall lära.

f7. I spelet “Black and White” kan spelarens husdjur växa up till olika monster beroende på hur djuret lär sig.

Nyare metoder börjar komma som en ghost AI som lär sig spelarens tekniker och kombinationer och försöker imitera dem. Exempel på detta finns i ”*Tekken Dark Resurrection*”, där spelaren kan spela in en match mot AI:n och ladda upp till andra spelare[2].

Problemet med denna typen av AI var att den lärde sig endast offline och alltså var byggd på föregående erfarenheter vilket gjorde att den inte kunde anpassa sig under spelets gång.

Detta gjorde den ej likt en mänsklig spelare eftersom den använde hela tiden samma spelstil vilket kunde utnyttjas precis som en statisk AI. Andra typer av learning: online-, offline- och supervised learning. Learning via direct adaptation eller indirect adaptation. Lista på spel med intressant använding av AI se referens [17

Syfte och mål

Målet var att använda tekniken ”Reinforcement Learning” för att utveckla en dynamisk och adaptiv AI, som lär sig under en match att öka och minska chansen att utföra ett kommando, beroende på hur den lyckades sist. För att med denna adaptiva AI:n kunna jämföra den mot en oadaptiv AI och utvärdera skillnaderna i spelbarhet. Försöka få svar på våra frågeställningar och huvudsakligen se vilken AI som presterar bäst och är roligast att spela mot. Förhoppningarna var att denna AI skulle prestera bättre än den icke adaptiva AI:n. Även fördjupa sig in i AI forskning som gått ifrån statiska “fuskande” AI till att förbättra AI implementationer med lärande och adaptiva metoder/tekniker. Vi använde oss av en färdig spelmotor för att snabbt kunna komma igång med att jobba med AI:n och inte lägga ner för mycket tid på spelmekaniken.

MUGEN

Motorn som vi använde var fightingspels motorn MUGEN. Elecbyte släppte en beta version 1999 och en full version släpptes 2001. MUGEN är "freeware" och låter vem som helst skapa och modifiera karaktärer. Dock var det väldigt svårt att skapa en AI och många omvägar fick göras för att få det att fungera. 2011 släpptes en ny version av MUGEN, som gjorde det lättare att skapa en AI, "*MUGEN 1.0*". MUGEN kan laddas hem här gratis på deras officiella hemsida[6].

f7. Fighting motorn och spelet MUGEN

Det finns oändligt många hemmagjorda karaktärer, banor och spel, baserade på MUGEN.
Bilden f7 visar "Kung Fu Man", en karaktär som man får med motorn.

Forskningsfråga

Vilken AI vinner mest mellan den adaptiva AI:n eller icke adaptiva AI:n?

För att testa detta gjorde vi två olika AI. En oadaptiv som slumpar mellan olika attacker och kombinationer och en adaptiv, som ökar chanserna för attacker och kombinationer beroende på hur den lyckades. MUGEN har ett spelläge där man kan se hur motorn simulerar en match mellan två AI. Här kunde vi kolla vilken som klarade sig bäst.

Vilken är roligast att spela mot mellan den adaptiva AI:n eller ikcke adaptiva AI:n?

För att besvara den här frågan gjorde vi ett formulär för speltestare att besvara, efter att de fått spela mot de båda AI. Med hjälp av formuläret kunde vi se vad spelarna tyckte om AI:n.

Beskrivning

Krav:

För att bygga vår AI behöver vi få reda på följande:

- Distansen mellan AI:n och spelaren i x och y led / p2bodydistx, p2bodydisty
- AI:n state / stateno
- Spelarens state /p2stateno
- Frekvensen, “Vikten” på handlingen (%) vilket var våran inlärningsmekanism / random < var(1)

Enligt [Manslow 2004] så är det också väldigt viktigt att applicera reinforcement learning vid rätt tillfälle. Därav så bygger vi en modell som använder reinforcement när någon karaktär blir skadad. Om spelaren blir skadad av AI:n så förstärks det beteendet som något positivt och om AI:n tar skada så minskar det beteendet och betraktas som något negativt.

Utförande

Med Reinforcement Learning skapade vi olika ”sekvenser” som innehåller olika kommandon tex sekvens1 innehåller block1, slag2, block2. För att få den adaptiv så ökade vi chansen att köra tex slag1 istället om slag2 missade. Detta skulle ge illusionen av att AI:n blir bättre under körnings tid.

Exempel på hur vi definierade ett kommando:

```
;AI Stand Light Punch
[State -1, AI Stand Light Punch]
type = ChangeState
value = 200
triggerall = p2statetype != C
triggerall = (ctrl) && (statetype = S)
triggerall = var(59) = 1
;----Combo from CLP----;
trigger1 = prevstateno = 400
trigger1 = p2movetype = H
;----Combo after Block----;
trigger2 = stateno = 150
trigger2 = movetype != H
;----Normal Punch----;
trigger3 = random <= 200
trigger3 = p2bodydist X < 100
trigger4 = hitcount < 3
trigger4 = p2bodydist X < 100
```

Här kan vi gå in via tre olika sekvenser. Om vi har varit i ”tillstånd 400” och fått in en träff på motståndaren (alltså om vi har gjort en ”Crouching light punch”), om vi har blockat motståndaren och inte tog en träff, eller om vi skall påbörja en sekvens.

Nyckelordet ”*triggerall*” förklarar vad som måste uppfyllas för alla sekvenser. Motståndaren får inte huka sig, samt AI:n måste stå upp och ha kontroll. Om det inte finns ett annat kommando med samma krav, kommer AI:n att gå in i tillstånd 200, om den uppfyller alla ”*triggerall*” kraven, samt kraven för en sekvens.

Följande exempel är ett exempel på hur vi gjorde när vi ville öka chansen för att gå in i tillstånd 440:

```
[State -1]
 type = varset
 var(58) = var(58) + 100
 trigger1 = prevstateno = 440
 trigger1 = p2movetype = H
 trigger1 = var(58) < 1000
 persistent = 0
```

Om AI:n har varit i tillstånd 440 och fått in en träff, skall variabel 58 ökas på med 100 så länge variabel 58 är under 1000. När vi sedan definierar vilka krav som skall uppfyllas för tillstånd 440 kan vi sätta att tex:

```
trigger1 = random <= var(58)
```

MUGENs random generator genererar ett tal mellan 0 och 999, så genom att öka värdet med 100, ökar vi chansen 10%

När vi tilldelar variablerna poäng med 100 kommer summan av variablerna snabbt att gå över 1000. Detta leder till att summan av alla variablerna kommer bli över 100%, och resulterar i att det inte kommer vara rätt procent chans att en visst slag körs. För att summan av alla variabler inte skall överstiga 1000, så måste vi normalisera variablerna.

```
fvar(25) = var(58) + var(57)...+ var(k)
fvar(24) = fvar(25) / 1000
```

Om variabeln fvar(25) kommer få en summa över 1000, till exempel 1200, kommer variabeln fvar(24) att bli 1.2. Vi vill komma så nära 1 som möjligt, och delar därför fvar(24) med sig själv.

```
fvar(24) = fvar(24) / fvar(24) = 1
```

För att det så skall bli rätt i båda leden, måste vi dela fvar(25) med fvar(24) alltså måste vi dela alla variabler fvar(24) för att det skall bli rätt.

```
var(58) = var(58) / fvar(24)
var(57) = var(57) / fvar(24)
...
var(k) = var(k) / fvar(24)
```

Detta fungerar också bra när summan av alla variabler blir under 1000 alltså att fvar(24) blir under 1. Att dela ett tal med ett annat under 1 kommer leda till att värdet ökar. Bara man inte delar ett tal med 0. För att undvika ett lokalt maxima bestämde vi oss även för att inte låta en attack överskrida max 90% dvs 900.

Problem och Lösningar

Då vi använde en färdig motor, fanns grundfunktionerna som till exempel, hoppa, gå och ducka. Dessa funktioner kördes igenom slumpmässigt och vi hade ingen kontroll över när AI:n skulle hoppa. Vi hittade dock en funktion för att AI:n inte skulle hoppa så ofta och lyckades reducera hopp frekvensen. Vi lyckades dock inte få AI:n att ducka lika ofta som vi hade velat. Denna koden finns inte i CMD filen, som vi definierade attackerna i, utan i CNS filen, där konstanterna är definierade.

Medan vi testade hur vår räknare fungerade, upptäckte vi att den alltid räknade för mycket. Detta berodde på att kraven som skulle uppfyllas, uppfylldes under flera tick. Vi löste det genom att lägga till ett krav som krävde att ett visst antal tick skulle ha gått efter att AI:n hade befunnit sig i ett visst tillstånd.

```
triggerall = time = 9
```

Då vi har alla våra variabler som heltal förutom fvar(25) och fvar(24) som var flyttal, blev normaliseringen inte riktigt 100% utan mellan 97,6% och 99,8%. Dock tyckte vi att vi kom tillräckligt nära för att godkänna det.

f8. Debugen i MUGEN lät oss se hur nära vi värdet 1 vi kom.

Ett problem som kvarstår är att vi ville lägga till en variabel för att blockera. Denna skulle inte fungera på samma sätt som attackerna. Vi ville att AI:n skulle blockera ju mer skada den har tagit, och minde för varje träff den får in på sin motståndare. Vi märkte dock att en attack variabel gick upp i ungefär 900 och sedan gick den plötsligt ner till 500. Trots att vi specificerade att värdet inte fick öka om det blev 900 eller över så gick den ändå över 1000. Vi visste inte vad detta berodde på, men tror att det var när blocken minskar så skall normaliseringen göra att alla attackerna skall öka. Men det förklarar inte varför variablerna sjunker drastiskt. Vi valde att inte använda någon variabel till blocken. MUGEN låter AI:n blocka med jämna mellanrum i alla fall. Men vi har inte kontroll över det.

Vi märkte att högersidan var den sidan som vann mest. Även om vi lät AI:n spela mot sig själv, så var det störst chans att högersidan skulle vinna. Detta tror vi är en bugg i motorn som vi inte kan göra någonting åt. För att lösa detta problemet gjorde vi testerna där varje AI fick börja på de olika sidorna lika många gånger.

Experimentdesign

För att besvara frågan om vilken AI som är roligast att spela mot bjöd vi in testare för att spela mot AI:n. Först fick de testa styrningen i ett tränings läge, där var AI:n avstängd. När spelaren kände sig redo fick spelaren möta, slumpmässigt, den adaptiva eller den icke adaptiva. När matchen var slut fick spelaren möta den andra AI:n. En match kan vara max tre ronder, det vill säga först till två vinster vinner. När båda matcherna var över fick spelaren besvara några frågor om vad de tyckte om AI:n. Vi döpte om AI:n till A och B där den adaptiva AI:n är A och den icke adaptiva är B. Detta gjorde vi för att spelaren inte skulle veta vilken som är adaptiv och på så sätt påverka deras svar om vilken som de tyckte var adaptiv.

Vi hade följande frågor:

1. Har du erfarenheter inom fightingspel?
2. Hur rolig var A att spela mot?
3. Hur rolig var B att spela mot?
4. Hur svår var A att spela mot?
5. Hur svår var B att spela mot?
6. Kände du att någon AI blev bättre under tiden som du spelade?
7. Kände du att någon AI var förutsägbar?
8. Tyckte du att någon AI betedde sig konstigt?
9. Övriga kommentarer.

På fråga ett var det en skala 1 till 5, där 1 var ingen erfarenhet och 5 var mycket erfarenhet. Anledningen till att vi tog med den här frågan var för att en person med mindre erfarenhet kommer att uppleva AI:n svårare. Frågorna 2 till och med 5 var också en skala 1 till 5, där 1 var tråkig och rolig, respektive lätt och svår. Fråga 6 och 7 fanns det svars alternativ A, B eller ingen. Frågorna 8 och 9 var en ruta där spelaren fick skriva fritt.

För att besvara frågan om vilken AI som vann mest mellan den adaptiva och den icke adaptiva, lät vi programmet göra jobbet. I MUGEN finns det en funktion som simulerar en match mellan två karaktärer eller i vårt fall, mellan två AI. Vi körde det tio gånger och bäst av tio vann. Vi delade upp så att var sin AI fick spela på vardera sida 5 gånger.

Resultat

Tabellen beskriver vad elva spelare tyckte om vardera AI.

Skala 1-5	**1**	**2**	**3**	**4**	**5**
Erfarenhet	2 st	3 st	2 st	3 st	1 st
AI A Rolig			4 st	5 st	2 st
AI B Rolig		2 st	3 st	5 st	1 st
AI A Svår		1 st		6 st	4 st
AI B Svår		2 st	5 st	3 st	1 st

Avvikelse mellan AI A och AI B beräknad ur T-test i ensidig test.
Rolig: 0,2453
Svår: 0,3103

Tabellen beskriver hur många av spelarna som tyckte respektive AI lärde sig under spelets gång och vilken AI som de tykte var förutsägbar.

	AI A	**AI B**	**Ingen**
Lärde sig	6 st	5 st	1 st
Förutsägbar	2 st	4 st	5 st

Avvikelse mellan AI A och AI B beräknad ur ett student T-test i tvåsidigt test.
0,8450

Tabellen beskriver antalet vinster respektive AI fick när de mötte varandra.

	Vänster	**Höger**	**Summa**
Adaptiv AI	5/5	4/5	9/10
Oadaptiv AI	1/5	0/5	1/10

Diskussion

Resultatet av formuläret visar att spelarna tyckte att den adaptiva AI:n var något roligare att spela mot än den icke adaptiva. Den adaptiva AI:n gav både vid observation och vid eget speltest mer motstånd än den icke adaptiva vilket vi tror resulterade i en roligare utmaning att försöka slå AI:n. Detta stämmer även överens med resultatet på frågan över hur svår vardera AI var, där den adaptiva var ett snäpp högre i svårighetsgrad än på den icke adaptiva. Den visade sig även vinna över den icke adaptiva 9 av 10 gånger i AI mot AI testet.
Det var svårt för testarna att se vilken AI som lärde sig där resultat av frågan i formuläret visar en ungefärlig splitt mellan båda AI. Detta kan bero på att vardera AI gav tillräckligt motstånd för att undvika utnyttjandet av eventuella kryphål. Den adaptiva AI:n var enligt spelarna även något mindre förutsägbar men ytterst lite vilket antagandet kan vara att båda AI's varierade sina attacker ungefär lika mycket fast i olika situationer. Den adaptiva blir mer förutsägbar då den får in många träffar på spelaren vilket gör att den har mindre chans att utforska nya strategier (eftersom strategin redan fungerar). Den icke adaptiva varierar sina sekvenser till viss del förutom i vissa fasta situationer. Test perioden var även kort vilket ger ett mindre tydligt resultat.

Problem som vi stötte på var att om den adaptiva AI:n blir väldigt mycket belönad att blocka mera kommer den att bli "dummare". Resonemanget till det var att den får mindre chans att utforska nya strategier på spelaren eftersom den väljer att blockera mera istället, även om den nu skulle blockera väldigt bra. Observation av testet visade även att den adaptiva AI:n använder i princip bara special attacker eftersom dem är betydligt svårare för spelaren att förhindra och därmed är dessa attacker mer effektiva att använda.

Även Thore Graepel, Ralf Herbrich och Julian Gold kom fram till i sitt arbete, "Learning To Fight"[14], att en lärande AI går att implementera ganska lätt i ett fightingspel. De lyckades skapa en lärande AI med hjälp av SARSA algoritmen. De hade även en belönings funktion där de tog hänsyn till de båda spelarnas liv. Leo Lee besvarade också frågan om vilken AI som är roligast att spela mot, i sitt arbete "Adaptive Behavior for Fighting Game Characters".[11] Även han fick positivt resultat vad gäller den adaptiva AI:n. Han byggde sin AI: utifrån "*Alpha Fighter*" spelet och den visade sig ge mer motstånd än en oadaptiv AI.

Framtida arbete

Om vi hade haft mer tid på oss så hade vi kunnat få den adaptiva AI:n att få mer poäng, inte bara beroende på om slaget träffade eller missade, utan beroende på hur mycket skada slaget gör och hur öppen AI:n blir efter ett slag. Detta är också delar som en spelare tar hänsyn till när han eller hon väljer en attack att göra. Andra alternativ är att låta AI:n gradvis anpassa sig till spelarens kunskaper och försöka hålla sig till en lagom balanserad nivå istället för att spela så bra som möjligt. Människor spelar inte lika perfekt som en AI som gör special attacker med hundraprocentigt utförande och vill helst ha en AI som kan besegras. Man kan även att låta AI:n lära sig av spelaren[12] och imitera spelarens olika combos och sekvenser för att spara undan dessa. AI:n kan sedan sätta vikter på hur bra sekvenserna fungerar. Detta skulle verka mera att AI:n liknar en riktig spelare då även "stå stilla" är något som en spelare gör.

Om vi hade kunnat mer om MUGEN när vi började arbetet, hade vi kunnat använda flyttal till alla variabler som skulle få poäng. Detta skulle låta oss få ett mer exakt värde efter normaliseringen.

Om vi hade byggt motorn själva, hade vi haft mer kontroll på AI:n, men det hade krävt att vi hade haft mycket mer tid till projektet även att sätta sig in i öppen källkod hade tagit förmodligen mer tid. MUGEN hade redan färdiga funktioner som att hoppa, gå, ducka och blocka. Detta har begränsat vårt arbete då det inte stämmer riktigt med hur stor chans det är att AI:n gör ett visst slag. Ibland kan AI:n hoppa när man inte vill och på så sätt kan den se väldigt dum ut. Men att AI:n gör dumma misstag kan få den att bli mer mänsklig. MUGEN lät oss också lägga ner all vår tid på AI:n och vi slapp tänka på grundliga delar som att gå, hoppa, ducka och blockera. Därför är vi väldigt tacksamma för MUGEN trots dessa brister. Arbetet gick trots allt ut på att bygga en AI och inte en motor.

Referenser

[1] Antonio Ricciardi & Patrick Thill - ”Adaptive AI for Fighting Games” December 12, 2008 @stanford.edu
http://cs229.stanford.edu/proj2008/RicciardiThill-AdaptiveAIForFightingGames.pdf
URL senast besökt 2012-07-02.

[2] Worapoj Thunputtarakul & Vishnu Kotrajaras - ”Data Aanalysis for Ghost AI Creation in Commercial Fighting Games”, GAMEON 2007 pp 37-41, EUROSIS, 2007. http://www.cp.eng.chula.ac.th/~vishnu/gameProg/papers/wpj_vishnu07.pdf
URL senast besökt 2012-07-02.

[3] Peter Dayan & Christopher JCH Watkins - ”Reinforcement Learning”, *Encyclopedia of Cognitive Science* (Wiley), 2012.
http://www.gatsby.ucl.ac.uk/~dayan/papers/dw01.pdf
URL senast besökt 2012-07-02.

[4] Abhijit Gosavi - ”A Tutorial for Reinforcement Learning” - Department of Engineering Management and Systems Engineering - Missouri University of Science and Technology - 219 Engineering Management, Rolla, MO 65409 - September 30, 2011
http://web.mst.edu/~gosavia/tutorial.pdf
URL senast besökt 2012-07-02.

[5] Florentin Woergoetter and Bernd Porr (2007) Reinforcement learning. Scholarpedia, 3(3):1448., revision #91704
http://www.scholarpedia.org/article/Reinforcement_learning
URL senast besökt 2012-07-02.

[6] http://www.elecbyte.com/mugen
URL senast besökt 2012-07-02.

[7] Pieter Huberdto Morarie Spronck - ”Adaptive Game AI”, ISBN 90-5278-462-0, 20 Maj, 2005. Tryckt av Datawyse b.v.. Maastricht, Nederländerna
http://ticc.uvt.nl/~pspronck/pubs/ThesisSpronck.pdf
URL senast besökt 2012-07-02.

[8] "Heavyweight Champ." *Wikipedia, The Free Encyclopedia*. Wikimedia Foundation, Inc. 2004. Juli 2, 2012.
http://www.enotes.com/topic/Heavyweight_Champ
URL senast besökt 2012-07-02.

[9] James Wexler - “Artificial Intelligence in Games: A look at the smarts behind Lionhead Studio's "Black and White" and where it can go and will go in the future,” - University of Rochester, NY 14627, Maj 7, 2002
http://www.cs.rochester.edu/~brown/242/assts/termprojs/games.pdf
URL senast besökt 2012-07-02.

[10] Mamakis Georgios - “Survey on Game AI Theory and Algorithms” - Technological Educational Institute of Crete and University of Glamorgan - Juli 18, 2011
http://cacgd2011.teicrete.gr/sites/default/lectures/MAMAKIS/IP_TEI_2011.ppt
URL senast besökt 2012-07-02.

[11] Leo Lee - “Adaptive Behavior for Fighting Game Characters” - The Faculty of the Department of Computer Science - San Jose State University – 2005. *Master's Theses.* Paper 2716. http://scholarworks.sjsu.edu/etd_theses/2716/
(http://www.cs.sjsu.edu/faculty/pollett/masters/Semesters/Fall04/leo/CS299Report.pdf)
URL senast besökt 2012-07-02.

[12] Sarayut Lueangrueangroj & Vishnu Kotrajaras - “Real-Time Imitation Based Learning for Commercial Fighting Games”. CGAT 09, International Conference and Industry Symposium on Computer Games, Animation, Multimedia, IPTV, Edutainment and IT Security, Amara Hotel, Singapore
http://www.cp.eng.chula.ac.th/~vishnu/gameProg/papers/CGAT_Real%20Time%20Imitaion%20Based%20Learning.pdf
URL senast besökt 2012-07-02.

[13] http://www.geekyblogger.com/2007/03/artificial-intelligence-in.html
URL senast besökt 2012-07-02.

[14] Thore Graepel, Ralf Herbrich, Julian Gold - “Learning to Fight” 2004 International Conference on Computer Games: Artificial Intelligence, Design and Education
http://research.microsoft.com/pubs/65639/graehergol04.pdf
URL senast besökt 2012-07-02.

[15] David M. Bourg, Glenn Seeman – “AI for Game Developers”, Juli 2004**,** O'Reilly Media
ISBN: 0-596-00555-5, pp. 135 – “Kung Fu Fighting”

[16] Andreas Pfeifer - ”Creating Adaptive Game AI in a Real-time Continuous Environment using Hierarchical NeuralNetworks” - Department of Computer Science
Knowledge Engineering Group, Mars 2009
http://www.ke.informatik.tu-darmstadt.de/lehre/arbeiten/diplom/2009/Pfeifer_Andreas.pdf
URL senast besökt 2012-07-02.

[17] http://www.pha.jhu.edu/~ggaspar/physics/games.html
URL senast besökt 2012-07-02.

[18] Andrade, Gustavo and Ramalho, Geber and Corruble, Vincent (2005) "Challenge-Sensitive Action Selection: an Application to Game Balancing". In *International Conference on Intelligent Agent Technology*, pp. 194--200.

www.cin.ufpe.br/~dlf2/01565536.pdf

URL senast besökt 2012-07-02.

[19] http://sites.google.com/site/ushyangproject/project-definition
URL senast besökt 2012-07-02.