Kandidatarbete i Medieteknik, 30 hp
Vårtermin 2013

Utvecklande AI

En studie i hur man skapar ett system för lärande AI

Mattis Axelsson
Sara Larsson

Handledare: Jonas Svegland & Peter Giger
Examinator: Lena Trojer
Blekinge Tekniska Högskola
Sektionen för planering och mediedesign

Sammanfattning

AI är något som blir allt viktigare inom dagens spel och får allt högre krav på att agera mänskligt och intelligent. Detta kandidatarbete undersöker vilken metod som är att föredra för att skapa en AI som kan lära sig av sina tidigare erfarenheter. Några av de metoder som undersöks är trädstrukturer, Artificial Neural Network och GoCap. Genom att skapa en applikation med en av metoderna samt göra en undersökning på hur AI:n i applikationen upplevdes fick vi resultat om denna metod var användbar. Utifrån detta diskuteras det ifall andra metoder hade varit mer effektiva, hur man hade kunnat förbättra AI:n samt hur framtiden för spel-AI skulle kunna se ut.

Nyckelord: AI, spel-AI, Artificiell Intelligens, trädstrukturer, Behavior Tree

Abstract

AI is something that has become more important in today's games and gets higher pressure to act human and intelligent. This thesis examines which methods are preferred when creating an AI that can learn from its previous experiences. Some of the methods that are examined are tree structures, Artificial Neural Network and GoCap. By creating an application with one of the methods and a survey of how the AI in the application was perceived we got a result that showed us if the method was functional. From this we discuss if the other methods would have been more effective, how we could have improved the AI and what the future for game-AI holds.

Keywords: AI, game-AI, Artificial Intelligence, tree structures, Behavior Tree

Förord

Detta är ett examensarbete utfört vid Blekinge Tekniska Högskola under våren 2013. Arbetet är på kandidatnivå och omfattar 30 högskolepoäng. Vi vill tacka våra handledare Peter Giger och Jonas Svegland samt alla medlemmar i vår handledningsgrupp för stöd och feedback under arbetets gång.

Ordlista

Agent – En autonom entitet som styrs av AI.

AI (Artificiell Intelligens) – Är skapad intelligens som strävar efter att förstå intelligenta system, i detta fall spel.

Algoritm – En algoritm är en uppsättning av väldefinierade instruktioner för att lösa en uppgift inom matematiken eller datavetenskapen.

Animation – Är när man skapar flera sekvenser till ett objekt för att skapa en rörlig bild.

Applikation – Är en typ av datorprogram som hanteras av en användare. En applikation kan bland annat vara kontorsprogram som kalkylprogram, kommunikationsprogram, e-postprogram och webbläsare. Det kan även vara nöjesprogram som datorspel.

Datorspel – Spel som spelas med hjälp av en dator. Datorn kan vara en persondator, en spelkonsol eller en mobiltelefon.

Dynamisk AI – En form av AI som kan förändras medans den körs, i motsats till statisk AI.

Felsökning – Är en metodisk process för att hitta och reducera buggar i ett datorprogram.

GUI (Graphical user interface) – Ett grafiskt användargränssnitt för att underlätta interaktionen mellan människa och program.

Neuron – Eller nervcell, är en celltyp i nervsystemet som är ansvarig för mottagandet och överförandet av nervimpulser.

NPC – Non-Player Character (icke spelbar figur), är en figur som inte styrs av en människa i ett datorspel.

Spelmotor – Är ett program som hanterar en del av mekaniken i ett spel. Bland annat kan den innehålla funktioner för grafik och fysik.

Innehåll

1. Inledning

Spel är ett medium som växer allt snabbare i dagens samhälle och i samtida spel är AI något som i stort sätt är ett obligatoriskt inslag. Allt högre krav ställs på spelen och även på AI:n, spelarna vill få samma utmaning från AI:n som från en människa. Detta gör det intressant att undersöka hur man kan skapa ett system för en AI som har mänskliga attribut, till exempel att den kan lära sig av det som händer. Därför ska vi i denna studie undersöka hur man på bästa sätt kan göra detta.

Vi kommer i denna text ha med en del engelska ord och uttryck eftersom många metodnamn är engelska, alternativt att det inte finns någon motsvarande svensk översättning.

2. Problemområde

2.1 Bakgrund

AI (Artificiell Intelligens) är något som människor länge har funderat över. Buchanan (2005, 53) nämner att man kan titta tillbaka på kända verk som Trollkarlen från OZ från 1900 (Baum 2012) och Frankenstein från 1818 (Shelley 1994) för att se tydliga inslag av AI. Även om man kan fundera över om Frankenstein verkligen är AI och inte bara en återuppväckning av en död människa visar detta och Baums mekaniska man Tiktok att det är något som har intresserat och inspirerat människor sedan länge.

Om vi ser tillbaka till vår barndom var scenariot att inte ha någon att leka eller spela spel med inte helt ovanligt. Oavsett om det var fotboll eller fia med knuff har nog de flesta någon gång haft problem med att hitta en motståndare som ligger på samma nivå som en själv, eller att ens hitta en motståndare över huvud taget. Ett sätt att lösa dessa problem var att göra spelen intelligenta och därmed se till att man alltid hade intelligent motstånd, med andra ord AI (Charles et al. 2008, 9).

I och med detta började tankarna om AI spridas till datorspel, detta ledde till att Pong, ett av de första och mest kända spelen med AI skapades av Al Alcorn och Nolan Bushnell 1972 (Charles et al. 2008, 17). Därifrån har AI:n utvecklats i snabb takt och det dröjde inte länge innan spel som testade nya saker inom AI skapades. Creatures (1996), Halo (2001) och Black and White (2001) är exempel på spel som blev uppmärksammades för sin nyskapande AI (Champandard 2007). Från dessa har tekniken fortsatt framåt och då spelen blir mer och mer komplicerade ställs det även högre krav på AI:n, den ska höja spelarens upplevelse och klara av de uppgifter spelet kräver. Detta har lett till att dagens AI är komplicerad och liknar människors beteende mer och mer, ibland kan det till och med vara svårt att se skillnad. Baum och Shelleys visioner kanske inte är långt borta?

Då AI numera i stort sätt är ett obligatoriskt inslag i spel och målet ofta är att få den att agera som en människa finns det ett stort intresse i att undersöka AI. Dock är det inte vanligt att AI i spel kan lära sig, men detta anser vi är något som kommer att slå stort inom en inte allt för lång framtid då AI:n utvecklas i snabb takt och spelarna förväntar sig att den ska vara mer trovärdig. Vilket gör det intressant och relevant att undersöka vilka möjligheter det finns att göra en AI som kan lära sig av sina erfarenheter.

2.2 Syfte

I detta arbete kommer vi att undersöka vilka metoder det finns för att skapa en AI för spel som är medveten om sina tidigare erfarenheter och kan ta dessa i beräkning inför framtida beslut. Utifrån denna undersökning kommer vi att skapa en applikation med en av metoderna för att se hur komplicerat det är att skapa ett system med de egenskaper som vi nämnde ovan och hur svårt det skulle vara att anpassa detta till ett spel.

2.3 Frågeställning

Frågeställningen lyder: Hur skapar man ett system för en AI där handlingar värdesätts olika beroende på tidigare omständigheter eller erfarenheter?

3. Tidigare forskning

I denna del av arbetet kommer vi först att kort beskriva vad AI generellt är. Efter detta stycke kommer vi mest att fokusera på att beskriva olika metoder som har potential att skapa en AI som kan ta del av sina tidigare erfarenheter. Sist kommer vi att diskutera och motivera fram till en metod som vi kommer ta vidare och göra en gestaltning av för att testa hur man skapar en AI som kan lära sig av sina erfarenheter.

3.1 AI

Det AI handlar om är att skapa program som är kapabla till att göra intelligenta uppgifter liknande de som människor och djur kan göra. Nu finns det avancerade program inom många områden såsom informationssökning och sortering. Medan utvecklingen har haft större problem inom områden där AI:n till exempel kan ta egna beslut och tänka kreativt, då det är svårare att ta fram algoritmer för dessa beteenden (Millington 2006, 4).

AI brukar främst delas upp i två olika områden, akademisk AI samt spel-AI. Det finns stora skillnader mellan dessa, men även likheter. De största skillnaderna är att de som jobbar inom akademisk AI är mer psykologiskt och filosofiskt inriktade, vilket leder till att de arbetar mer med att tillverka AI med syfte att hjälpa oss i vår vardag, till exempel att överföra tal till skrift. Spel-AI däremot inriktar sig mot att förhöja upplevelsen i spel, genom att till exempel få NPCer att bete sig intelligent och därmed skapa en utmaning för spelaren. Det finns dock undantag där områdena samarbetar, ett exempel på detta är militärsimulatorer. Där kombineras spelelement med mer seriösa inslag för att skapa en verklighetstrogen simulator som kan användas för att träna militären (Millington 2006, 4-9). Områdena har olika möjligheter och problem, till exempel har ofta den akademiska AI:n stora resurser när det gäller hårdvara och tid medans spel-AI är mer begränsad inom de områdena (Tozour 2002, 5-8). Då detta arbete inriktar sig på spel-AI kommer resterande delar fokusera på det området.

Enligt Buckland (2005, XX) är AI illusionen av intelligens, om spelaren tror att agenten är intelligent är den också det. För att få AI:n att verka intelligent behöver det inte innebära att man måste skapa stora komplexa system, detta är något som även Tozour (2002, 10) tar upp. Tozour säger att det är dumt att AI kallas för AI då intelligens är ett subjektivt ord och AI i spel inte alltid behöver vara intelligent på det sätt som människor ofta förknippar ordet med. Det finns ett ordspråk som speglar detta bra ”Om en fågel ser ut som en anka, simmar som en anka och låter som en anka, så är fågeln förmodligen en anka”. Ifall AI:n beter sig som en anka kommer även spelaren att uppfatta den som en sådan, oavsett hur avancerat systemet är.

Dagens AI i spel har enligt Millington (2006, 9) tre basbehov, dessa är förmågan att kunna förflytta agenter, att kunna fatta beslut om vart de ska förflyttas samt att kunna tänka taktiskt och strategiskt. Hur dessa utförs kan dock variera stort beroende på vilka andra behov det finns i spelet.

3.2 Metoder

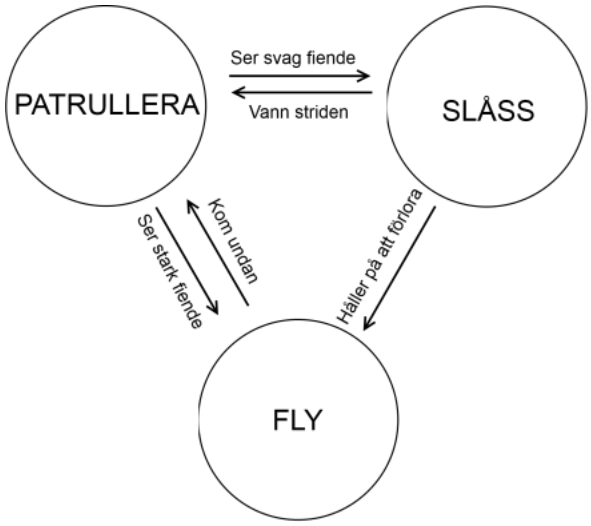
I detta avsnitt går vi igenom olika metoder som är intressanta för vår frågeställning. Vi har valt metoderna för att de antingen är vanligt förekommande i spel, till exempel Finite State Machines, eller att de är intressanta för att göra avancerad AI som kan utvecklas och lära sig, till exempel Artificial Neural Networks.

3.2.1 Finite state machine

Finite State Macines (FSM) är och har länge varit en av de vanligaste metoderna för att programmera AI i spel och troligtvis kommer de att finnas kvar länge. Anledningen till detta är att de är enkla och snabba att programmera, enkla att felsöka samt lätta att förklara för personer som inte är insatta i programmering (Buckland 2005, 43). FSMs är främst användbara i spel där AI:n ska bete sig på ett begränsat antal sätt och kan bli svåröverskådliga ifall AI:n blir för avancerad, det kan också vara svårt att göra mer dynamisk AI då FSMs är ganska statiska (Millington 2006, 318). Dock kan de vara en bra grund för mer avancerade system som Fuzzy Logic och Artificial Neural Networks (Buckland 2005, 44).

En FSM fungerar genom att agenten har olika lägen den kan befinna sig i och det är bara möjligt för den att befinna sig i ett läge åt gången. Vanligtvis är handlingar eller beteenden bundna till läget och medan agenten befinner sig i det läget kommer den att utföra dessa. Lägena är länkade genom övergångar som har olika villkor och när villkoren har uppfyllts byter agenten läge (Millington 2006, 318).

Ett exempel Millington (2006, 319) tar upp är en agent som ska bete sig som en vakt, de lägena den har är att patrullera, slåss och fly. Figuren nedan visar ett exempel på hur en FSM för denna typ av agent skulle kunna se ut.



Figur 1: Beskrivning av en Finite State Machine

3.2.2 Fuzzy logic

Människor har en stor förmåga att använda uppskattningsord, till exempel att något är nära eller långt borta, utan att definiera dessa ord har vi ungefär samma uppskattning om vad de betyder. Detta är dock något som datorer har problem med eftersom de vill ha fasta värden för att kunna mäta det. Om vi säger att nära är 0-2 meter, mittemellan är 2-4 meter och långt borta är mer än 5 meter, ifall ett föremål då är 4,9 meter bort är det enligt datorn på ett mittemellan avstånd, även om det för en människa skulle uppfattas mer som att det är långt borta, det är detta problem Fuzzy Logic försöker att lösa (Buckland 2005, 415-416).

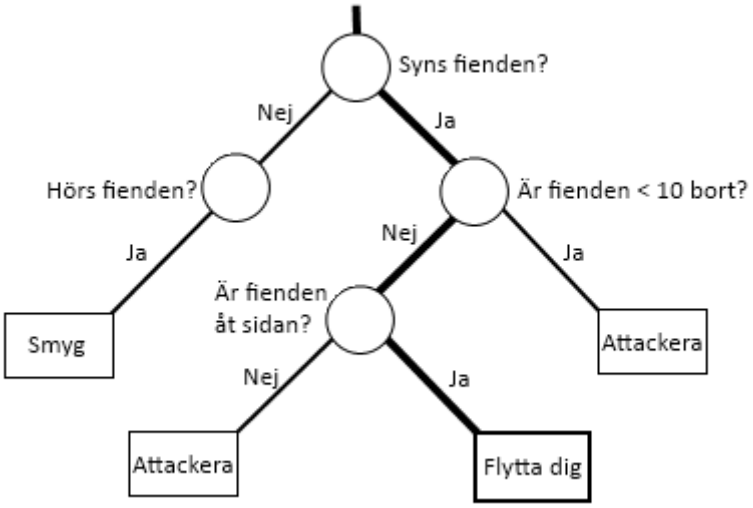
För att skapa Fuzzy Logic använder man sig av Fuzzy Sets, dessa kan liknas med Finite State Machines lägen, skillnaden är att agenten kan befinna sig i flera set samtidigt, den kan också befinna sig olika mycket i olika set. Oftast ger man agenten ett värde mellan 0 och 1, dessa kallar man grad av medlemskap, där 0 innebär att den inte är i det setet medan 1 innebär att den fullständigt är i setet.

Då Fuzzy Logic bara fungerar med grad av medlemskap krävs det konvertering för att kunna applicera detta i spel som oftast hanterar ren data, detta kallas fuzzification, när man konverterar tillbaka datan kallas det defuzzification. Detta gör att man kan göra linjerna mellan en agents beslut mindre tydliga. Till exempel om agenten ska smyga för att den är rädd och gradvis blir modigare kan man genom Fuzzy Logic göra detta synligt istället för att den helt plötsligt slutar smyga för att den växlade över till läget att vara modig som hade varit scenariot om en FSM använts (Millington 2006, 344-345).

3.2.3 Trädstruktur

Trädstrukturer är uppbyggda på ett sådant vis att de består av anslutna beslutspunkter. Trädet har ett startbeslut, vilket fungerar som grund och som sedan är ansluten till en serie av andra beslut den kan välja mellan. Varje beslut som görs är baserad på trädets kunskap om omgivningen. Dessa beslut fortsätter att göras tills det inte finns några mer beslut att ta. Vid varje beslut finns det en handling som utförs omedelbart och när denna handling är slutförd går den sedan till nästa beslut. Enligt Millington (2006, 593) finns det stor möjlighet att anpassa metoden till vad som krävs för spelet, med antingen mindre träd som ger mer generella beteenden eller större mer invecklade träd för att göra beteendena mer specifika.

Nedan är ett exempel på hur ett träd kan se ut när en agent ska bestämma vad den ska göra mot en fiende.



Figur 2: Beskrivning av en trädstruktur

Trädstrukturer är enkla att skapa och det finns även möjlighet att lägga in funktioner som ger AI:n möjlighet att lära sig av vad som händer. Detta genom att använda sig av algoritmer, en av de vanligaste inom spel är Quinlans ID3 algoritm, ID3 står för Inductive Decision Tree Algorithm 3 eller Iterative Dichotomizer 3. Algoritmen skapar ett träd dynamiskt från set av observationer och handlingar som görs under stark tillsyn.

3.2.4 Goal Oriented AI

Goal Oriented AI fungerar ungefär som en Finite State Machine, fast istället för att gå igenom olika lägen som innehåller logik för att kunna övergå till andra lägen arbetar Goal Oriented AI med en kollektion av hierarkiska mål. Enligt Buckland (2005, 379) finns det antingen atomiska eller sammansatta mål. Atomiska mål definieras av en enkel uppgift, beteende eller handling som att hitta en position eller ladda ett vapen. Sammansatta mål däremot är mål som består av ett antal delmål som i sin tur antingen kan vara atomiska eller sammansatta, vilket definierar en fast hierarki. Sammansatta mål brukar oftast vara mer komplexa uppgifter som att bygga en fabrik eller retirera och hitta skydd.

Denna hierarkiska struktur ger användaren en intuitiv mekanism för att definiera en agents beteende eftersom det delar många likheter med en människas tankesätt. Människor sätter upp abstrakta mål baserat på deras behov och bryter sedan ner dessa till mindre planer som de sedan följer, precis som en Goal Oriented AI.

Ett exempel på hur en Goal Oriented AI kan vara uppbyggd är att AI:ns mål är att köpa ett svärd. Detta delar den sedan in i mindre mål som kan vara gå till affären, ta reda på vad svärdet kostar, planera hur den ska få tag i pengar, planera vägen till pengarna, skaffa pengarna, planera vägen tillbaka till affären och tillslut köpa svärdet. Det som är bra med detta system är att ifall det skulle dyka upp något som förhindrar AI:n från att utföra sitt delmål, lägger den delmålet i en väntelista och utför sedan de nya delmålen för att kunna lösa förhindret. När dessa är avklarade tar den upp sitt undanplockade delmål och fortsätter.

Enligt Millington (2006, 380) kan man snabbt få förvånansvärt bra resultat med Goal Oriented AI, speciellt i spel där det bara finns ett fåtal handlingar AI:n kan utföra. En nackdel är dock att metoden inte kan ta i beräkning vilka sideffekter som en handling kommer att ha.

3.2.5 Artificial Neural Network

Vår hjärna är sammansatt av miljarder av neuroner**,** varav varje neuron är länkad till tusen av andra neuroner för att forma ett komplext nätverk. Artificial Neural Network (ANN) är en metod där man försöker härma vår hjärnas processorkapacitet genom att bygga ett system av dessa neuroner (Bourg 2004, 269). ANN erbjuder en del fördelar över de mer traditionella AI-teknikerna som till exempel Finite State Machines då det gör det möjligt för utvecklaren att simplifiera dessa system genom att förvisa viktiga beslut till en eller flera neuroner, även kallat noder. Systemet har också potentialen att anpassa sig medan det körs.

Trots dessa fördelar används ANN inte ofta i spel, anledningen till detta är att dess användning i spel är begränsad. De två största anledningarna är att ANN är exceptionell på att hantera icke linjära problem, vilket inte är lätt med traditionella metoder. Detta gör dock ibland att det är svårt att förstå exakt vad AI:n gör och hur den kom fram till sitt beslut. Den andra anledningen är att det ibland kan vara svårt att förutse vad metoden kommer generera för resultat, speciellt om AI:n är programmerad att lära sig och anpassa sig medan den körs. Dessa två anledningar gör det relativt svårt at testa och felsöka ANN jämfört med att felsöka en Finite State Machine (Bourg 2004, 269-271).

Enligt Charles et al (2008, 21-23) kan man dela in ANN i två olika lägen, ett läge där den använder sig av sina förprogrammerade kunskaper samt ett läge där den organiserar de vanligaste valen från dessa kunskaper, det sistnämnda kallas Learning Mode. Learning Mode kan bli indelat i tre olika kategorier: Supervised, Unsupervised och Reinforcment Learning.

3.2.5.1 Supervised Learning

Supervised Learning innebär att man säger åt AI:n vilket resultat man vill att den ska ge när den får in en specifik uppgift. Uppgiften blir sedan skickad framåt genom nätverket tills den når noden som hanterar resultatet som då aktiveras. Man kan sedan jämföra svaret AI:n räknade ut med svaret man ville ha. Ifall svaret stämmer behöver det inte ändras något i nätverket, om det är ett annat svar som räknas ut behöver man justera variablerna för att försäkra sig om att nätverket kommer ge rätt svar i framtiden när den får samma uppgift.

3.2.5.2 Unsupervised Learning

I denna typ av system finns det ingen extern källa som bestämmer svaret i förväg utan svaret är generellt baserat på information som är lokalt till varje nod. Detta brukar också hänvisas som självorganiserande, i den mening att nätverket självorganiserar sig baserat på de svar från uppgiften den blir presenterad med. Till skillnad från Supervised Learning vet man inte vilket resultat man kommer att få.

3.2.5.3 Reinforcement Learning

Denna metod avser att maximera resultatet genom att använda sig av ett slags trial and error-inlärning. För att nätverket ska kunna lära sig vet den inte vilken åtgärd den ska använda sig av utan måste upptäcka det själv genom att se vilken som ger bäst resultat. Om en åtgärd är lyckad justerar nätverket sina variabler för att förstärka detta beteende annars avskräcker systemet nätverket från att använda detta.

3.2.6 GoCap: Game Observation Capture

GoCap är en maskininlärningsteknik som tränar AI genom att låta den observera en människa när denne spelar. Agenten har tre olika lägen: styrd av en spelare, styrd av autonomiska kontroller och träningsläge. När agenten är i träningsläget styrs den av spelaren, precis som i läget där den är styrd av en spelare. Skillnaden är att den sparar ner vilka handlingar spelaren gör och detaljer om under vilka villkor dessa handlingar utfördes sparas. Beteenden definierar dessa villkor i form av regler och när agenten är i sitt autonomiska läge utgår den från dessa regler när den bestämmer vilken handling den ska utföra. Efter att agenten har lärt sig alla regler jämför den vilka handlingar den själv skulle välja med de handlingar som spelaren gör, när dessa stämmer överens ger den feedback till spelaren att träningen är klar. Detta kan också hjälpa till att hitta regler som blir triggade hela tiden och kanske måste formuleras om (Alexander 2002, 579-583).

Nackdelen med detta är att innan AI:n kan användas måste den tränas och det finns en möjlighet för spelaren att träna den på ett fel sätt vilket leder till att dess handlingar inte passar in med hur det var tänkt att den skulle bete sig. Alexander (2002, 585) anser att maskininlärningstekniker som denna kommer att ha stor påverkan på spel-AI i framtiden och göra det möjligt för mer avancerad och intelligent AI. Dock finns det problem, främst i minneshantering, som måste lösas innan detta blir möjligt.

3.3 Motivering till vald metod

När vi funderade över vilken metod vi ville ta vidare till produktionen var det några metoder som föll bort på en gång då de inte fanns möjlighet att implementera funktioner som kan få AI:n att lära sig av sina erfarenheter. Dessa metoder var Finite State Macines, Fuzzy Logic och Goal Oriented AI. Vi valde även att stryka GoCap då den inte hanterar inlärning på det sätt vi var ute efter, dessutom kände vi att den är för avancerad för den nivå vi befinner oss på.

Då återstod två metoder som vi vägde mellan, Artificial Neural Networks och en trädstruktur. Vi fann båda metoderna intressanta och kände att de hade möjlighet att hjälpa oss undersöka vår frågeställning. Metoderna hade sina för och nackdelar, ANN kändes som en passande metod eftersom det finns möjlighet att skapa en AI som lär sig under tiden applikationen körs, dock fanns det en risk att den var för avancerad för oss. Trädstrukturer däremot beskrivs som enkla att använda men har inte lika stor möjlighet att skapa AI som lär sig medan applikationen körs, vilket då kunde skapa problem för oss att nå ett resultat.

Vi träffade Dr. Johan Hagelbäck som är lektor inom AI för att se vad han hade för åsikter om de tankar vi hade. Han ansåg att vi skulle stryka ANN då den är bättre lämpad för metoder med mycket brus, till exempel ansiktsigenkänning. ANN är också alldeles för avancerad för att vi skulle hinna göra klart ett projekt inom den tidsram vi hade. När vi gick vidare till trädstrukturer ansåg han att det skulle passa till vårt syfte. Dock inte med ID3-algoritmen som vi tog upp, eftersom den inte används under tiden applikationen körs utan innan den startas, vilket inte passade för vårt projekt. Istället skulle man kunna lägga till prioritet eller sannolikhet till de olika noderna, som man sedan väger beroende på hur stor framgång AI:n hade när den utförde noden. På det sättet kommer den sakta att lära sig vilken väg i trädet som fungerar bäst.

Utifrån de synpunkterna vi fick från Hagelbäck bestämde vi att gå vidare med trädstrukturer då det känns som den bästa metoden för oss. Han bedömde också att vi skulle kunna hinna klart med att skapa en applikation med den metoden under den tidsram vi hade, vilket känns betryggande.

4. Tillvägagångssätt

Denna del av arbetet kommer gå igenom hur vi jobbade med vårt projekt samt varför vi valde att göra de val som gjordes. Det första stycket är döpt till val av gestaltning och tar upp hur vårt spel ser ut, sedan fortsätter det till vilken metod vi arbetade efter under gestaltningens gång. Efter detta går vi igenom hur gestaltningen utfördes, bland annat beskrivs det hur de viktigaste delarna i projektet fungerar. Tillsist går vi igenom hur undersökningen som utfördes för att få feedback på gestaltningen lades upp.

4.1 Val av gestaltning

4.1.1 Applikationen

För att knyta vår frågeställning till gestaltningen valde vi att tillverka en applikation som använder sig av AI i ett fightingspelskoncept. Anledningen till detta är att det på ett tydligt sätt demonstrerar hur AI:n reagerar och lär sig beroende på vilket sätt motståndaren slåss på. Då det är enkelt att bygga upp grunden för ett fightingspel ger det oss mer möjlighet att fokusera på att utveckla AI:n. Detta leder till att vi på ett bra sätt kan undersöka vår frågeställning eftersom vårt mål med applikationen är att bygga ett system som kan användas för att skapa en AI som lär sig av sin motståndare under tiden applikationen körs.

Det första som händer när applikationen startas är att en menyskärm visas, där finns det tre olika val för hur användaren kan interagera med AI:n. Antingen kan användaren möta AI:n själv, alternativt låta en offensiv eller defensiv förprogrammerad motståndare möta AI:n. Det finna även tre andra val, ett där det visas vilka som har gjort applikationen, ett där man kan läsa instruktioner för applikationen samt ett val att avsluta.

Om valet att möta AI:n väljs visas en ny scen där två får står vända mot varandra. Dessa får befinner sig i en arena och är redo att slåss. Det vita fåret är alltid AI:n som lär sig under stridens gång och det svarta fåret är användaren eller någon av de förprogrammerade motståndarna. På skärmen finns fem stycken attackrutor för varje får, varav rutan som fåret väljer att använda blir markerad. Varje får har en livsmätare som sänks beroende på hur mycket skada de tar, om någon av livsmätarna töms är striden över. För att tydligt visa vad som sker kommer det upp en ruta som beskriver en attack när användaren håller musen över ikonen, när en attack utförs kommer det upp hur mycket skada varje får tog för att användaren ska kunna se hur lyckade attackerna var.

Anledningen till att gestaltningen byggdes upp på detta vis var för att på ett simpelt men tydligt sätt visa hur AI:n lär sig. Att det finns olika sätt att möta AI:n på lades in för att kunna testa hur den reagerar mot olika motståndare. På detta sätt går det att se hur AI:n reagerar annorlunda mot olika typer av motståndare och därmed lär sig av vad som händer.

4.1.2 Diagram

När striden är över visas en ny scen där det finns ett diagram över vilka attacker som AI:n och dess motståndare använde sig av samt vilken runda attacken utfördes. Det visas också vem som vann, samt ett val att återvända till menyn eller att avsluta applikationen.

Diagrammet som visas i slutet är ett bra sätt för användaren att se hur AI:n reagerade under applikationens gång och gör det lätt att visa att AI:n lärde sig av det som hände. För att göra diagrammet tydligt för användaren valde vi att göra AI:ns staplar röda och dess motståndares staplar blå. Dessa staplar visar då vilken attack som utfördes under vilken runda. Det går också att stänga av en eller flera staplar för att kunna granska resultatet för enbart en attack eller fler om så önskas.

Eftersom resultatet hur AI:n betedde sig är viktigt för oss att tydliggöra då vår gestaltning kretsar kring detta valde vi också att lägga in en funktion som visar hur mycket varje attack användes i en procentuell skala. Detta gör det lätt att se vilken attack som användes mest under hela applikationens gång och ger då användaren extra data för att utvärdera vår AI.

4.2 Projektmetod

Vi valde att jobba med en prioriteringslista där vi skrev upp alla delar som behövdes för att applikationen skulle kunna nå upp till de mål som satts upp. Dessa delar numrerades sedan från ett till fem beroende på hur viktiga de var för att applikationen skulle fungera. Detta är inte en standardiserad projektmetod utan en del från Scrummetoden, prioriteringslistan kan jämföras med Scrums produktbacklog. Produktbackloggen fungerar genom att man rankar de uppgifter som behöver göras beroende på vad som är viktigast för produktionen och dessa utförs sedan i den ordningen (Schwaber och Sutherland 2011, 12-13). Anledningen till att vi bara tog denna del från Scrummetoden var att det passade bäst för vårt projekt samt för hur vi personligen tycker om att arbeta. I tabellen nedan kan ni se hur de olika delarna prioriterades.

Del av applikationen	Prioritering
Behavior Tree	1
Diagram	1
Grafik	2
Spelare	2
Enkel AI-bot	2
GUI	3
Ljud	4
Roterande kamera	4
Meny	4
Övriga effekter	5
Fungera på Android	5
Databas med frågor	5

Tabell 1: Prioriteringslistan vi använde under vår gestaltning.

Utifrån denna prioriteringslista gjordes ett schema med deadlines när de olika delarna skulle vara klara. Vissa saker behövde vara gjorda innan det gick att fortsätta med andra, därför gick det inte att ta dem helt i prioriteringsordning då det fanns fler saker som vägde in. Förutom prioritet behövdes det också tänkas på vad som skulle vara klart innan något annat kunde göras, hur lång tid sakerna tog att skapa samt när vi skulle få grafik från grafikerna. Utifrån dessa riktlinjer skapades schemat, som kan ses nedan. Tanken var att alla delar hade den veckan de ligger under som arbetstid, undantaget är Behavior Tree som hade två veckors arbetstid eftersom det var den största och mest avancerade delen i applikationen.

Fredag 29/3	Fredag 5/4	Fredag 12/4	Fredag 19/4	Fredag 26/4
Enkel AI-bot	Behavior Tree	Strid fungerar	Roterande kamera	Fungera på Android
Spelare	Meny	Diagram	Grafik	Databas med frågor
Sannolikhet	Meny med attacker		Applikationen fungerar	Övriga effekter
	Livsmätare		Ljud	Undersökning

Tabell 2: Schemat vi använde under gestaltningen.

Metoden vi valde att arbeta efter fungerade bra, vårt arbete stagnerade inte då det fanns fasta deadlines att sträva mot. Under arbetets gång insåg vi dock att det inte fanns tid för att hinna med både att göra en undersökning, databas samt att få applikationen att fungera på Android. Eftersom vi ansåg att undersökningen var viktigast och mest relevant för arbetet ströks de andra delarna.

4.3 Utförande av gestaltning

4.3.1 Motor

Vi började med att jobba med Java i Eclipse eftersom båda var intresserade av att lära sig Java. Dock märkte vi efter att ha suttit med det i lite mer än en vecka att det tog alldeles för lång tid att lära sig, då varken språket eller motorn gav oss några fördelar valdes detta bort.

Istället valdes Unity3D som är en 3D-spelmotor. Fördelarna med detta var bland annat att det inte behövde göras några metoder för att hantera grafik och animationer, det var redan inbyggt i motorn. I Unity3D går det att jobba i Javascript och C#, eftersom båda hade erfarenhet i C# valdes det språket då vi ville undvika att lägga mer tid på att försöka lära oss något nytt.

Unity3D finns i två olika versioner, pro som är en betalversion och innehåller fler verktyg, till exempel enklare sätt att hantera ljus och skuggor, med pro ingår det även tillstånd att släppa sina applikationer kommersiellt för att tjäna pengar. Det finns även en gratisversion som innehåller färre verktyg och ifall spelet släpps kommersiellt får man bara tjäna en viss summa innan man måste uppgradera till pro (Unity Technologies 2013). Vi valde att använda gratisversionen eftersom vårt syfte inte var att tjäna pengar, och de extra verktyg som vi blir utan hindrade oss inte från att kunna svara på frågeställningen.

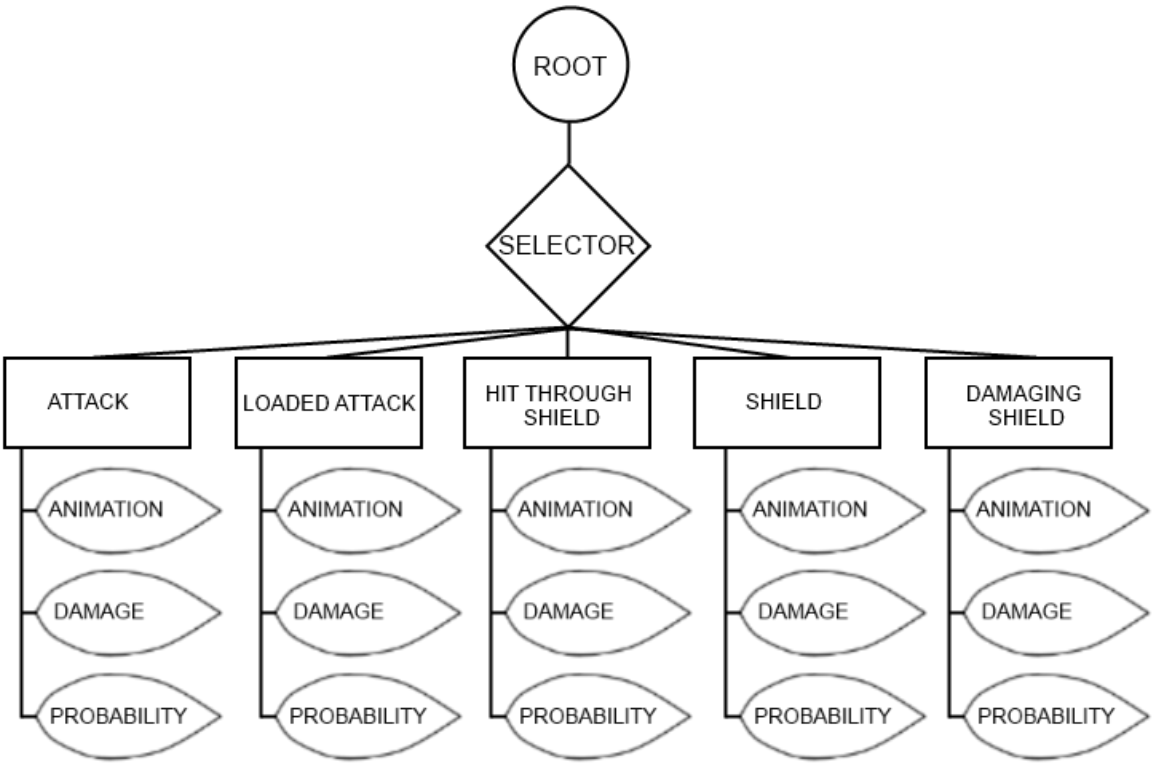
Då båda hade erfarenhet med att jobba i Unity3D och C# sedan tidigare var inlärningskurvan liten och det handlade mest om att friska upp minnet. Detta underlättade arbetet och gav oss bättre möjlighet att fokusera på det som var viktigast, att besvara vår frågeställning.

4.3.2 Behavior Tree

Det finns olika sorters trädstrukturer, de två vanligaste är Desicion Tree och Behavior Tree, skillnaden är att Desicion Trees är bättre på att hantera kedjor av långa beslut medan Behavior Trees är bättre på att hantera just beteenden. Därför ansåg vi att ett Behavior Tree passade bättre för vår gestaltning, eftersom vi kommer att jobba en del med beslut men mer med beteenden.

När systemet för vårt Behavior Tree skulle skapas började vi med att läsa hur Behavior Trees är uppbyggda. Bland annat var Champandards artiklar (2007, 2012) och Knaflas blogginlägg (2011) givande för att öka förståelsen för Behavior Trees och det var från dessa vi tog grunden till vårt system.

Systemet är uppbyggt av noder och varje nod har olika syften att fylla. Det finns fyra olika sorters noder, Selector, Sequence, Leaf och Decorator. Med dessa går det att bygga upp många olika beteenden för en AI. Selector är den nod som tar beslut, den bestämmer vilken väg i trädet som ska följas. Dessa beslut kan tas på olika sätt, till exempel genom slumpning, prioritet eller sannolikhet. Vi har valt att använda sannolikhet då vi kan ändra den under tiden applikationen körs för att få AI:n att lära sig vilka beteenden som är fördelaktiga att använda. Sequence har flera barnnoder, dessa kör den i den ordning de ligger i och kan användas för att bygga upp beteenden. Leafs är de noder som ligger sist i trädet, precis som löven på ett träd. De kan inte ha några barnnoder och innehåller funktionalitet för beteenden. Den sista typen av nod är Decorators, de används för att dekorera andra noder, till exempel för att hindra ett beteende att köras för ofta eller begränsa hur många gånger ett beteende får köras.



Figur 3: Hur det Behavior Tree som används i vår applikation är uppbyggt

4.3.3 Sannolikhet

AI:n bestämmer vilka beteenden som ska utföras genom att använda sannolikhet, i och med detta behövdes det skapas olika metoder för att den skulle kunna hantera sannolikheten på rätt sätt. De metoder som behövdes var att AI:n skulle kunna ändra sannolikheten, samt att den skulle kunna välja en attack utifrån hur stor sannolikhet attacken hade.

När AI:n startas har alla attacker lika stor sannolikhet att användas, utifrån detta väljer den en attack att utföra. När attacken har utförts kommer systemet räkna ut hur lyckad attacken var och med hur stort värde sannolikheten för attacken ska modifieras med, beroende på detta kommer sannolikheten för att attacken används igen ökas eller sänkas.

Då det finns fem olika attacker skapades en lista med fem element som representerar attackerna. Den totala sannolikheten för alla element skulle alltid ligga på 100, anledningen till detta var att sannolikheten använder sig av procent, och får inte överstiga 100 %. Detta innebar att alla element kommer att ha värdet 20 när applikationen startas. När en attack utförts ändras sannolikheten för den attacken samtidigt som värdet på de resterande elementen modifieras med en fjärdedel av värdet som attacken modifierades med för att det totala värdet ska fortsätta vara 100.

Ett exempel för att visa hur det fungerade är om en attacks sannolikhet ökades från att vara 20 till 21 ändrades alla andra attackers chans att utföras till 19.75 för att behålla det totala värdet på 100. Det sattes också upp en säkerhetsåtgärd som gjorde att alla attackers sammanlagda värde alltid blev 100.

AI:n använder sig av den kumulativa sannolikheten för att välja vilken attack den ska utföra. Detta fungerar genom att listan med elementen för attackerna sorteras från lägst till högst värde, efter detta genereras ett slumpmässigt tal mellan 0 och 100 som sedan jämförs med den kumulativa sannolikheten för varje element. Den kumulativa sannolikheten räknas ut genom att sannolikheten för ett element i listan adderas ihop med sannolikheten från de element som ligger före den. Efter detta väljer AI:n det element som ligger närmast det slumpmässiga talet och utför den attacken.

4.3.4 Grafik och ljud

Vi valde att göra vår applikation i 3D eftersom det gav oss mer frihet att bygga upp en miljö samt att vi kunde arbeta mer med ljussättning och atmosfär, jämfört med om applikationen gjorts i 2D. Då Unity3D även är mer fokuserad på att jobba med just 3D kändes detta som det bästa valet för vår gestaltning.

Då ingen av oss hade kunskaper i att skapa grafik valde vi att fråga utomstående om hjälp. Vi bad Jesper Olofsson att göra modellen för karaktärerna och dess animationer, och Robert van den Born för miljön samt GUI och andra 2D texturer. Vi var oroliga för att grafiken inte skulle bli klar eftersom de inte hade möjlighet att jobba med det mer än på fritiden, dock hade vi problem med att hitta en alternativ lösning.

Det blev snabbt problem då Olofsson meddelade att han tyvärr inte hade tid att animera karaktärerna. Vi funderade på hur vi skulle kunna lösa detta och efter diskussion med vår mentor Jonas Svegland fick vi klartecken att använda en modell som skolan äger, vilken redan hade färdiga animationer.

Då vi också saknar kunskap i att skapa ljud så vände vi oss även här till utomstående. Vi frågade Cajsa Larsson om ljudeffekter och vände oss till ArenaNet som är spelföretaget bakom bland annat Guild Wars 2. Cajsa gav oss de ljudeffekter vi behövde, och ArenaNet svarade att eftersom det inte är ett kommersiellt projekt får vi använda oss av deras musik, så länge vi uppger att ArenaNet LLC har copyright.

4.4 Undersökning

Vi valde att göra en undersökning då vi ville få feedback och synpunkter på vår AI från personer som inte hade varit inblandade i projektet, detta eftersom vi då kunde få en mer subjektiv syn på hur vår AI upplevdes.

Vi använde oss av Googles formulär då det var gratis och enkelt att jobba med, formuläret är gjort på engelska för att kunna nå ut till fler personer. Den målgrupp vi valde för vår undersökning var personer som är intresserade av spel och AI eftersom det är dessa personer som vårt arbete är riktat mot. Dessa kunde nås genom att posta vår undersökning på BTHs Facebook grupper då det är ett snabbt sätt att nå ut till studenter på vår skola, där de flesta är intresserade av spel. Vi gjorde även ett foruminlägg på aigamedev.net som är en sida för AI-utveckling, på så sätt kunde vi nå ut till personer som är intresserade av, och kanske även jobbar med AI.

De frågor som ställdes var:

- What/which game mode(s) did you try?
    - AI vs. Player
    - AI vs. Offensive bot
    - AI vs. Defensive bot
- Did it seem like the AI learnt what attacks worked best or just choose by random?
    - 1-10
- What was your opinion of the AI's behavior?
    - It choose attack by random
    - It learned which attack that worked best
    - It knew beforehand what attack to use
- Is this a good way to present an AI that learns during play?
    - Yes
    - No
- If no, why not?
- Do you have experience with AI development?
    - Yes
    - No
- Other opinions

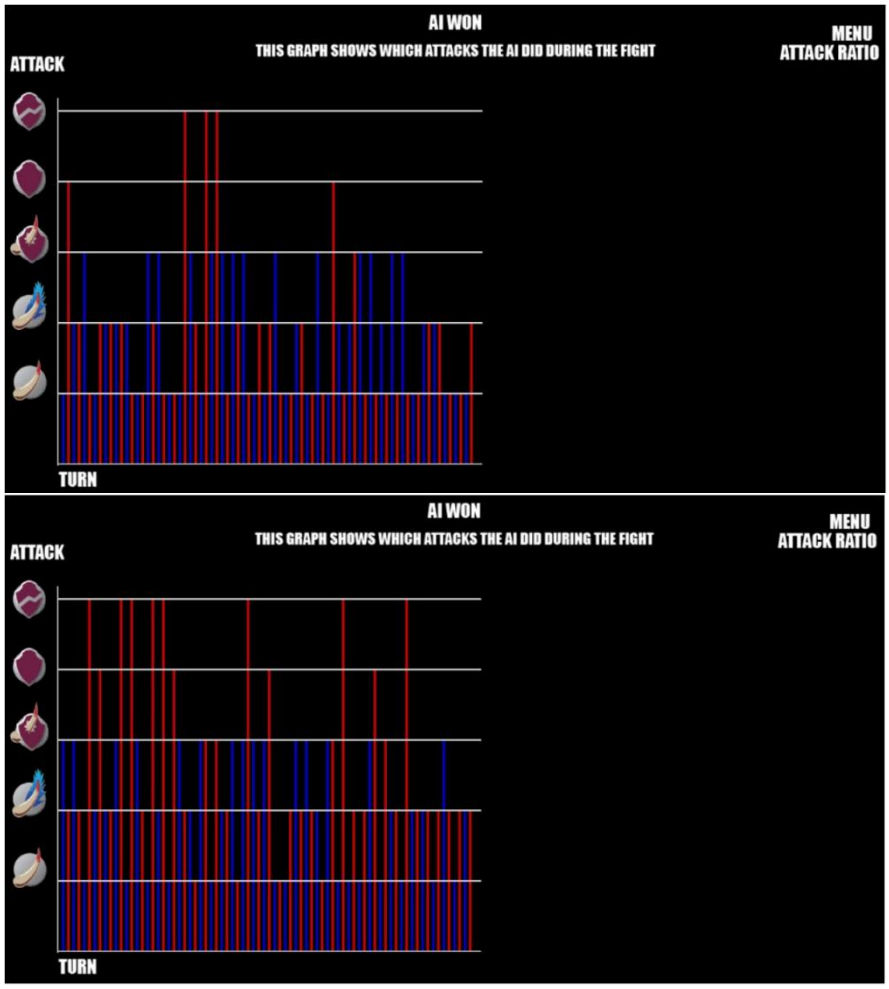
Från dessa frågor kunde vi se hur olika testpersoner upplevde vår AI, vi kunde också jämföra olika grupper med varandra, till exempel mellan de som är erfarna med att jobba med AI och de som inte är det. Detta var intressant för att vi då kunde se ifall en persons kunskap om AI påverkade hur de upplevde den i vår applikation.

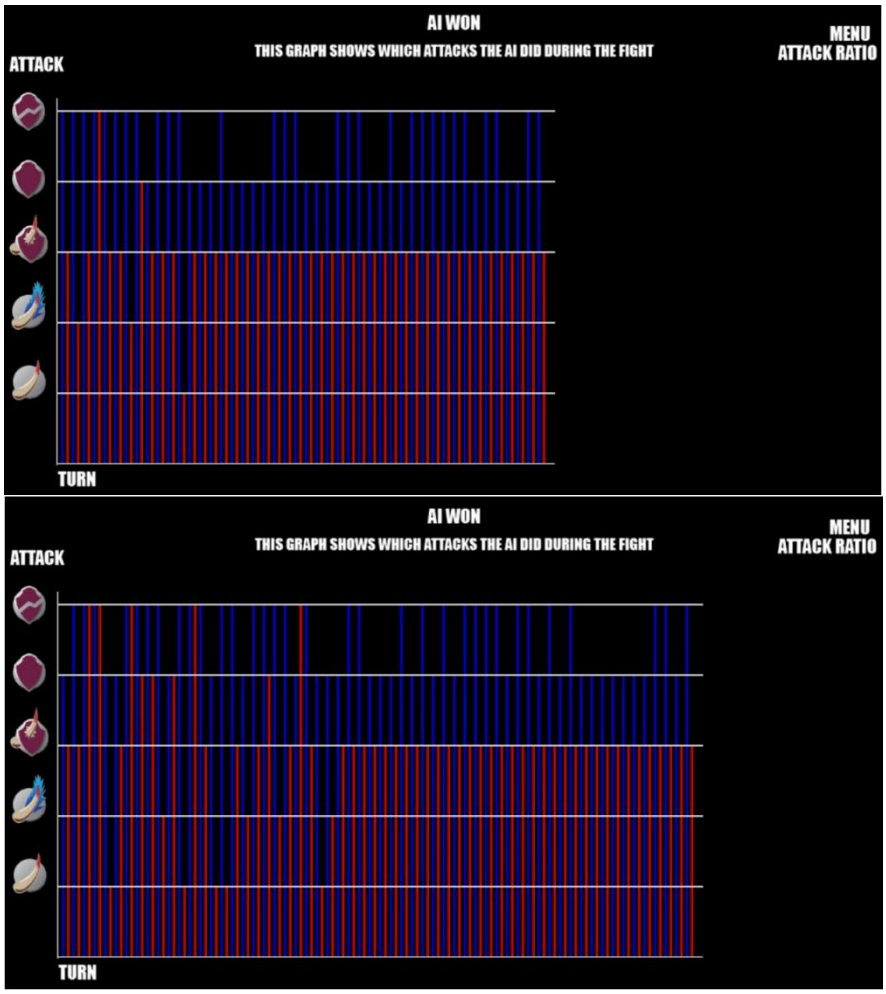
5. Resultat och diskussion

I denna del kommer resultaten vi fått att redovisas, dels från den applikation vi har skapat samt från undersökningen som utfördes. Vi kommer även göra en kort tillbakablick på hur resultatet blev, ifall det nådde våra förväntningar eller inte.

5.1 Applikationen

Då vår frågeställning var att ta reda på hur man kan skapa ett system för en AI som ska kunna värdesätta sina handlingar beroende på tidigare händelser och erfarenheter byggde vi en applikation där AI:n styrs av ett Behavior Tree, ett system som kan användas för detta ändamål. När applikationen testades gick det tydligt att se att AI:n anpassade sig efter det som hände under applikationens gång. AI:ns handlingar varierade beroende på vad dess motståndare gjorde, detta kunde vi tydligt se via diagrammet som skapades i slutet av applikationen. Nedan kan ni se fyra diagram som visar AI:ns och dess motståndares handlingar under olika spelomgångar.





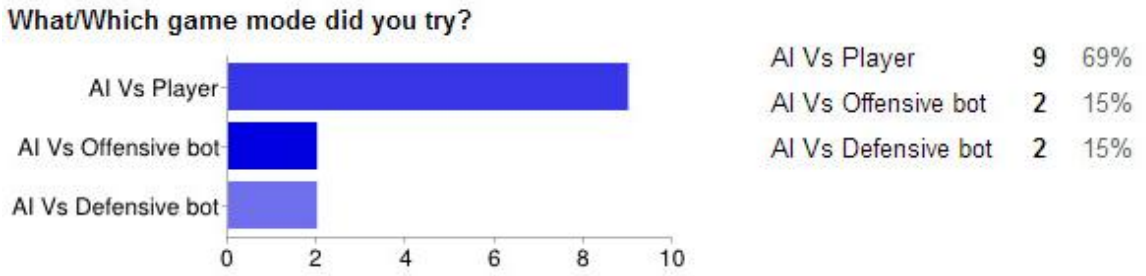
Figur 4: Diagram från vår applikation som visar vilka attacker AI:n(röd) och dens motståndare(blå) gjorde under applikationens gång. De två första bilderna visar AI:n som möter en offensiv motståndare. De två sista bilderna visar AI:n som möter en defensiv motståndare.

Här kan vi se att vårt system fungerar mot förprogrammerade motståndare som följer en konsekvent spelstil, men fungerar AI:n mot mänskliga motståndare? För att ta reda på detta valde vi att göra en undersökning där vi lät andra personer testa vår applikation och svara på några enkla frågor, denna enkät går att läsa mer om under nästa kapitel. Resultatet som vi fick fram från detta var att AI:n delvis fungerar mot mänskliga motståndare, men att den saknar en del finness för att kännas helt användbar. Det som flera påpekade var att AI:n inte har någon mönsterigenkänning, det vill säga att om spelaren gör en attack tre gånger och sedan byter till en annan och fortsätter med detta i ett mönster upptäcker inte AI:n mönstret. Detta leder till att AI:n inte känns lika mänsklig eftersom mönster är något som människor upptäcker snabbt.

Utöver detta fungerar AI:n som vi planerade i början av vår gestaltning, vilket vi känner oss nöjda med. Något som förvånade oss när vi arbetade med applikationen var att det var relativt lätt att bygga vårt Behavior Tree, medan det svåra låg i att balansera applikationen, dels vilka värden attackerna skulle ha, men också hur snabbt sannolikheten justerades. Detta var något som vi inte förutsåg i början av gestaltningen och som kom som en positiv överraskning, då vi hade planerat att vårt Behavior Tree skulle ta upp mycket av vår produktionstid. Detta gav oss då mer tid att lägga på just balansering, vilket var viktigt för att få det resultat vi var ute efter.

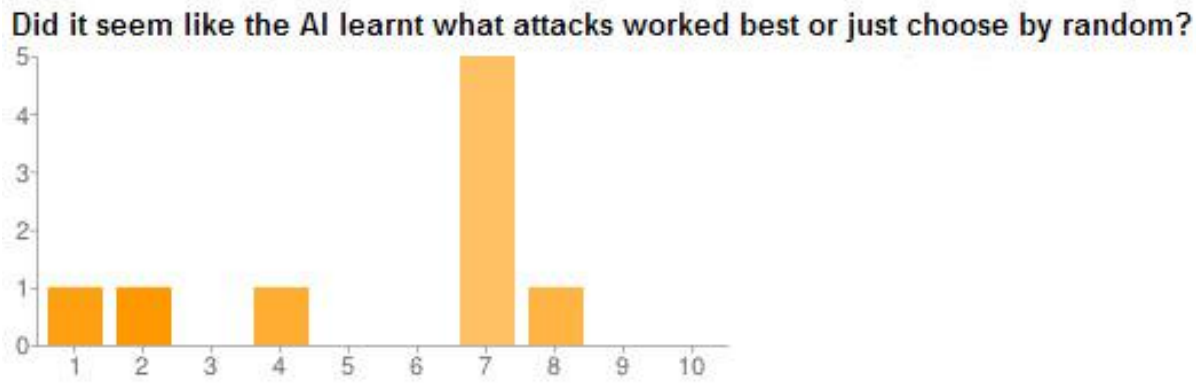
5.2 Undersökning

Eftersom vi enbart kunde ha vår undersökning i en vecka var det bara nio personer som svarade på vår undersökning. Även då det var få svar kände vi att de som svarade hade lagt ner mycket tid och tanke bakom svaren. Detta gjorde att vi fick bra kvalitet på undersökningen och kunde bortse från antalet som svarade.



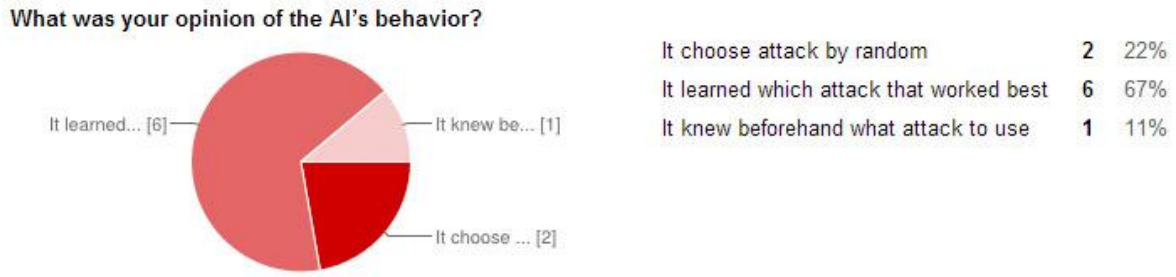
Tabell 3: Resultat från vår undersökning på frågan ”What/which game mode did you try?”

Här kan vi se att alla som gjorde undersökningen åtminstone testade att själva spela mot AI:n. Vilket visar att det är det som människor finner mest intressant, detta var också bra då det är just den delen av applikationen vi ville undersöka.



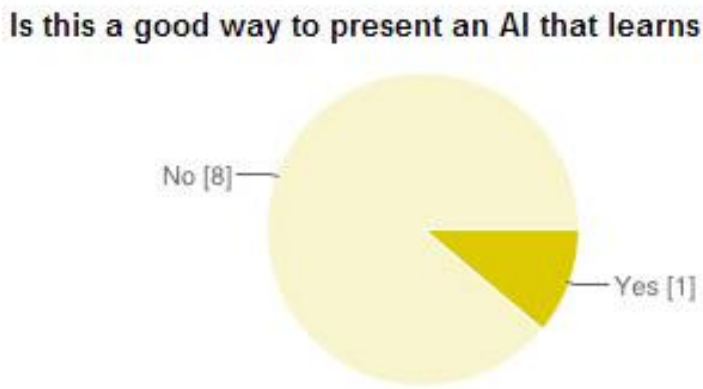
Tabell 4: Resultat från vår undersökning på frågan ”Did it seem like the AI learnt what attacks worked best or just choose by random?”

Denna graf visar hur lärande AI:n uppfattades av de som gjorde undersökningen. Till vår glädje tyckte de flesta att den lärde sig vilka attacker som fungerade bäst och inte bara valde attacker slumpmässigt.



Tabell 5: Resultat från vår undersökning på frågan "What was your opinion of the AI's behavior?"

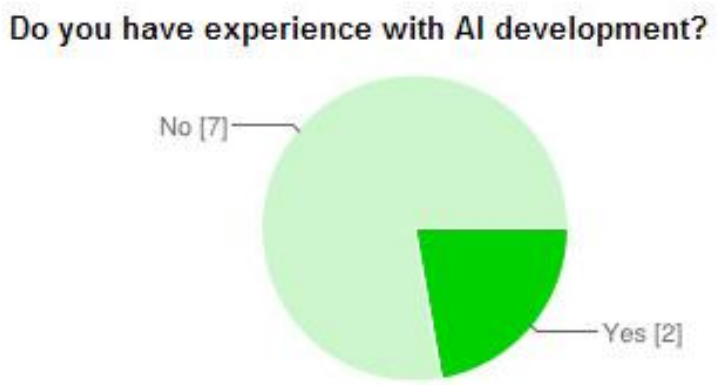
Även detta diagram hjälper oss att se att de flesta som testade vår applikation uppfattade AI:n på det sätt som var menat. Dock syns det att det finns saker som vi kan förbättra eftersom alla inte svarade att den lärde sig vad som fungerade bäst, ett svar som vi tog åt oss av var: *"There is a risk that the AI will appear to just choose attacks by random, which is not what you're after. But it seems like you have a good start here. To make it more 'human' you could add pattern recognition since that is something humans use a lot."* Som vi nämnde tidigare skulle detta kunna hjälpa vår AI att uppfattas som mer mänsklig.



Tabell 6: Resultat från vår undersökning på frågan "Is this a good way to present an AI that learns during play?"

Till vår förvåning tyckte de flesta inte att det här var ett bra sätt att presentera vår AI på. Detta diagram följdes upp med en följd fråga där vi ville att de skulle beskriva varför detta inte var ett bra sätt att presentera AI på.

Många ansåg att applikationen var otydlig och förstod inte vad som hände, detta var något som vi tog åt oss av och förbättrade applikationen så att det är lättare för användaren att följa med i vad som händer. En annan åsikt som återkom var att mönsterigenkänning skulle hjälpa för att få AI:n att fungera bättre, detta är något som vi i nuvarande skede inte kommer att implementera, men som kan vara intressant att fundera över om vi skulle vidareutveckla applikationen.



Tabell 7: Resultat från vår undersökning på frågan "Do you have experience with AI development?"

Här ser vi att de flesta som svarade på vår enkät inte hade någon erfarenhet i att jobba med AI. Detta är något vi kommer att ta i beaktning i diskussions delen då svaren kan skilja sig åt beroende på om de hade erfarenhet med AI sedan tidigare eller inte.

5.3 Tillbakablick av resultat

Efter att ha tillverkat vår applikation samt fått feedback och kritik på den färdiga produkten kände vi oss nöjda. Tack vare undersökningen fick vi chans att förbättra applikationen då vi själva inte hade sett det som påpekades. När vi tittade tillbaka på hur det var tänkt att vår applikation skulle se ut och hur den blev tycker vi att resultatet blev bättre än vad vi hade hoppats. Vi har en annan grafikstil som gör applikationen snyggare än det som var tänkt från början, det är också mer innehåll i applikationen än vad som var planerat, till exempel diagrammet i slutet som fungerar ypperligt för att förklara vad som händer under applikationens gång. AI:n som var den viktigaste delen av arbetet blev vi också nöjda med, den blev som planerat och det finns möjlighet för förbättring och vidareutveckling.

Med hjälp av applikationen kan vi tydligt visa att vårt Behavior Tree-system fungerar för att skapa en AI som kan lära sig av sina tidigare erfarenheter. Detta får vi stöd för både från undersökningen men även genom att göra speltester själva genom att låta AI:n möta förprogrammerade motståndare.

6. Slutdiskussion

Som avslutning kommer vi att diskutera och dra slutsatser från de resultat vi fick, samt diskutera kring vår process. Hur vi hade kunnat ta vårt arbete vidare och hur vi hoppas att framtiden kommer se ut för spel-AI kommer också tas upp.

6.1 En fungerande metod för att skapa lärande AI

Resultatet vi fick fram visar på att metoden vi använde för vår undersökning fungerar samt att den speglar vår forskning, även då vi enbart använde oss av Behavior Trees och inte någon av de andra metoderna. Då vi enbart använde oss av en metod och inte undersökte de andra kan vi inte helt forskningsmässigt säga vilken av alla metoder som är att föredra för just detta ändamål. De andra metoderna är intressanta i sin utsträckning och hade troligtvis även de kunnat ge oss ett intressant resultat, men det hade varit tvivelaktigt om vi hunnit med att skapa en bra produktion för att visa upp dessa metoder. Detta leder till att vi anser att Behavior Tree är den bästa metoden för just denna undersökning och ändamål.

Då vårt resultat blev bättre än planerat samt att tillverkningen av applikationen gick snabbare än förväntat, känns det som att vi valde en bra metod för vår frågeställning. Om vi ser tillbaka på den diskussion som vi hade med Johan Hagelbäck då han föreslog att användandet av Behavior Tree skulle gynna vårt resultat bäst hade han rätt. Behavior Tree var den metod som passade vårt arbete, och var också något som föll inom vår tidsram. Då vi även kan se ytterligare möjligheter med metoden, bland annat med mönsterigenkänning tycker vi att detta är en metod som kan användas för ändamålet att få en AI att lära sig av sina erfarenheter.

Om man tittar tillbaka på vår undersökning som utfördes ser vi att resultatet vi nådde var positivt eftersom 67 % av alla som svarade ansåg att AI:n valde sina handlingar beroende på vilken attack den lärde sig fungerade bäst. Vi tror att denna siffra skulle ökat om vi hade gjort undersökningen i skrivantes stund då vi utifrån den feedback vi fick gjorde om applikationen till det bättre. Detta går dock inte att bevisa eftersom det inte finns tid att göra om undersökningen för att få ett resultat av detta.

6.2 Slutkommentar

För att nå ett ännu bättre resultat tror vi att det hade varit fördelaktigt att lägga en del av forskningsmomentet i att undersöka hur människor ser på AI och intelligens, samt hur de anser att en AI ska fungera för att uppfattas mänskligt. Detta eftersom det är en viktig del när man bygger en AI, att få den att verka mänsklig. Som vi nämnde i kapitlet om AI, ett system behöver inte vara avancerat för att AI:n ska uppfattas som mänsklig, det viktiga är att bygga upp en illusion om att den är intelligent för då kommer den även att uppfattas som intelligent. Det är något som vi tydlig kunde se under produktionen, som vi nämnde tidigare var systemet för vårt Behavior Tree inte svårt att bygga, utan utmaningen låg i att designa hur AI:n skulle bete sig samt att balansera den. Därför hade forskning om själva designandet av en AI, och inte bara vilka metoder man kan använda för att bygga den varit fördelaktigt i vårt arbete.

Ytterligare en sak som skulle ha kunnat hjälpa oss att skapa en mer mänsklig AI hade varit att testat AI:n på utomstående personer under produktionens gång och diskuterat med dem hur de uppfattade AI:n samt vad de skulle vilja se annorlunda för att uppfatta den som mer mänsklig. Detta eftersom man lätt kan bli ”hemmablind” för sitt eget arbete och ha svårt för att upptäcka brister, något som vi fick erfara när vi utvärderade resultaten från undersökningen. Eftersom AI är en finkänslig sak att skapa tror vi att det är viktigt att hela tiden ha utomstående ögon på den för att underlätta produktionen och för att skapa en AI som uppfattas som intelligent.

När vi tittar djupare i undersökningen kan vi se att det är viss skillnad i svaren beroende på om personen hade tidigare erfarenhet med AI eller inte. Den största skillnaden låg i att de som hade erfarenhet med AI gav oss förslag på hur vi direkt kan förbättra AI:n, medan de som inte hade någon erfarenhet gav oss tips på hur vi kan förbättra själva applikationen. Detta beror troligtvis på att de som inte har erfarenhet av AI inte kan ge oss tips på att förbättra den delen eftersom de inte förstår hur det fungerar, vilket leder dem till att ge mer generella förslag. En annan skillnad var att flera av de som inte hade erfarenhet fann det svårt att förstå hur applikationen fungerade och vad som hände. Vi anser att det är extra viktigt att lyssna på de som inte har erfarenhet av AI då majoriteten av de som spelar spel inte har det, även om det kan bli en mer intressant diskussion med personer som har erfarenhet. Därför valde vi att lägga in fler element som gör applikationen enklare att förstå, till exempel beskrivningar av vad attackerna gör samt vilka attacker de är bra mot. Även då detta inte var relevant för vår frågeställning anser vi att det är viktigt att alla kan ta del av vår produktion. I och med detta tror vi att fler kommer uppskatta vår produktion och förstå vilka val AI:n gör, vilket kommer att leda till att de tycker den är mer intelligent och mänsklig.

Hela 89 % tyckte inte att vi presenterade AI:n på ett bra sätt, och även de som hade tidigare erfarenhet med AI hade den åsikten vilket fick oss att fundera över hur man hade kunnat presentera det bättre. Vi tror dock att de missuppfattade denna fråga, eftersom istället för att ge förslag på en annan presentation gav de förslag till att förbättra det vi redan hade. Vilket får oss att tro att de tyckte att presentationen var bra, men med en del brister, vilka de ville påpeka. Nu när vi har åtgärdat dessa brister tror vi att vi skulle få en mer positiv feedback på denna fråga.

6.3 Framtiden

Vad har då framtiden att säga oss gällande AI? Förhoppningsvis kommer de tyngre och mer avancerade metoderna att bli mer lättillgängliga allteftersom datorerna utvecklas med mer prestanda samt att metoderna utvecklas för att bli mer inriktade mot spel. Precis som Alexander (2002, 585) nämner hoppas även vi att maskininlärningstekniker som GoCap och ANN kommer ha större påverkan på spel-AI eftersom de öppnar upp för mer avancerad och intelligent AI.

Då vi enbart skapade applikationen för vårt kandidatarbete kommer vi inte att vidareutveckla den. Ifall vi hade haft mer tid eller planer på att jobba vidare med detta hade nästa steg varit att implementera mönsterigenkänning då vi, och även många av de som svarade på vår undersökning, känner att detta skulle kunna göra vår AI mer mänsklig.

6. Källförteckning

Alexander, T., 2002. *GoCap:* Game Observation Capture, in: *AI Game Programming Wisdom*. Charles River Media, Hingham, Mass, ss. 579–585.

Baum, L.F., 2012. *The wonderful Wizard of Oz*. Barnes & Noble, New York.

Bourg, D.M., 2004. *AI for game developers*, 1st ed. ed. O'Reilly, Sebastopol, CA.

Buchanan, B., 2005. A (Very) Brief History of Artificial Intelligence. *AI Magazine*, vol. 26, ss. 53–60.

Buckland, M., 2005. *Programming game AI by example*. Wordware Pub, Plano, Texas.

Champandard, A., 2007. *Top 10 Most Influential AI Games* [WWW Dokument]. AiGameDev.com. URL http://aigamedev.com/open/highlights/top-ai-games/ (hämtad 1.31.13).

Champandard, A., 2007. *Understanding Behavior Trees* [WWW Dokument]. AiGameDev.com. URL http://aigamedev.com/open/article/bt-overview/ (hämtad 3.28.13).

Champandard, A., 2012. *Understanding the Second-Generation of Behavior Trees* [WWW Dokument]. AiGameDev.com. URL http://aigamedev.com/insider/tutorial/second-generation-bt/ (hämtad 3.28.13).

Charles, D., Fyfe, C., Livingstone, D., McGlinchey, S., 2008. *Biologically inspired artificial intelligence for computer games*. Medical Information Science Reference, Hershey, PA.

Knafla, B., 2011. *Introduction to Behavior Trees* [WWW Dokument]. #AltDevBlogADay. URL http://www.altdevblogaday.com/2011/02/24/introduction-to-behavior-trees/ (hämtad 3.28.13)

Millington, I., 2006. *Artificial intelligence for games*, The Morgan Kaufmann series in interactive 3D technology. Elsevier, Amsterdam ; Boston : Morgan Kaufmann.

Schwaber, K., Sutherland, J., 2011. *Scrumguiden, Den definitiva guiden till Scrum: Spelets regler* [WWW Dokument]. Scrum.org. URL http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum%20Guide%20-%20SE.pdf (hämtad 4.11.13).

Shelley, M.W., 1994. *Frankenstein, or, The modern Prometheus*. Penguin, London.

Tozour, P., 2002. The Evolution of Game AI, in: *AI Game Programming Wisdom*. Charles River Media, Hingham, Mass, ss. 3–15.

Unity Technologies, 2013. [WWW Dokument]. https://store.unity3d.com/ (hämtad 5.14.13).