Aarhus Universitet
Datalogisk Institut
Åbogade 34
8200 Århus N

27. maj 2004

Robotspil

Introduktion af fysiske adaptive autonome agenter som del af interaktive underholdningsspil.

Sune Kristensen, årskort 19962098

The Three laws of robotics

1. Robots must never harm human beings or, through inaction, allow a human being to come to harm.
2. Robots must follow instructions from humans without violating rule 1.
3. Robots must protect themselves without violating the other rules.

— Isaac Asimov

Forord

I forbindelse med udarbejdelsen af mit speciale er der en række personer, der har bidraget til processen på den ene eller anden måde. Jeg vil derfor gerne sige tak til:

Ole Caprani: lektor og specialevejleder for god vejledning konstruktiv kritik, halvsort humor og for at have ladet mig hjælpe med til ved en række spændende projekter.

Peter Buch Jakobsen: medstuderende og kontorfælle. God sparringspartner og har været med til at højne moralen under processen.

Christian Ulrik Andersen: ph.d. studerende i computerspil for godt samarbejde i forbindelse med spilkursus og indledende Spybot forsøg samt for gode råd til opgaveskrivning.

Klaus Testrup fra Jydsk Pædagog-Seminarium i Randers, børn 0.B på Katrinebjergskolen samt Martin og Sebastian fra 3. klasse på Århus Friskole.

Anders Jensen og Michael Barratt Andersen fra LEGO for at stille Spybots og ekspertise til rådighed.

Microsofts XBOX afdeling: specielt Christian Rasmusen for udlån af XBOX.

Jesper Andreas Hansen, der har hjulpet med at stramme op på specialets struktur.

Sidst, men ikke mindst tak til min kone for at rette mine stave- og kommafejl samt mine fire børn for at undvære deres far så mange dage og aftener.

Udarbejdelsen af dette speciale har været en langvarig men udbytterig og til tider endda morsom proces.

— Sune Kristensen

Indhold

Figurer

Tabeller

1 Indledning

Inden for de senere år er prisen på robotter faldet markant. Derfor bliver der i dag solgt robotter som legetøj med kommunikationsudstyr; udstyr der tidligere var forbeholdt forskere inden for militæret og universitetsverdenen.

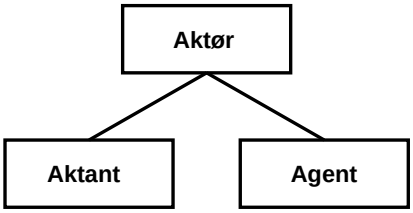
Den billigere avancerede teknologi åbner nogle helt nye muligheder inden for spilverdenen. Hvor det tidligere var nødvendigt at beskæftige sig med virtuelle agenter på en computerskærm for at opnå en interaktiv spiloplevelse, er det nu muligt ved hjælp af robotter at flytte interaktive spil ud på gulvet i børneværelset, på græsplænen i haven eller måske endda i vandkanten på stranden i stedet for. Muligheden er dog ikke blevet udnyttet i særlig høj grad, når der ses på mængden af udbudte robotspil i forhold til udbudet af computerspil.

I løbet af 2002 og starten af 2003 havde jeg nogle interessante oplevelser og erfaringer med et kommercielt robotprodukt fra LEGO kaldet **Spybotics**. Målgruppen for Spybotics er børn fra 9-13 år. Produktet består af en lille robot, en **Spybot**, der kan styres med en fjernbetjening eller gives et simpelt kontrolprogram, der udformes i et grafisk programmeringssprog. Erfaringerne og oplevelserne bestod i at benytte Spybotics produktet anderledes end det umiddelbart lægger op til, nemlig til at konstruere et fysisk spil inspireret af computerspil.

Dette speciale omhandler processen med at konstruere en robotspilsprototype ved hjælp af LEGO Spybotics produktet.

Undervejs vil aktør, aktant og agent blive benyttet som begreber til at beskrive og forstå computer- og robotspil.

Jeg anvender aktør som samlet betegnelse for de figurer, der optræder i computerspil og de robotter der optræder i robotspil. Der bliver skelnet mellem to typer aktører; nemlig aktanter der betegner de aktører, der kontrolleres af en person og agent som betegner autonome aktører.



FIGUR 1.1: Oversigt over forholdet mellem aktør, agent og aktant

I et computerspil som Pacman er spøgelserne agenter og osten er aktant. Forholdet mellem aktør, aktant og agent er illustreret i figur 1.1.

1.1 Erfaringer med LEGO Spybotics

I efteråret 2002 konstruerede jeg sammen med Christian U. Andersen et robotspil ved hjælp af Spybots. De første spæde ideer til dette projekt opstod efter afprøvningen af dette robotspil. Derfor er obervationerne af spilscenarierne ikke videnskabeligt bygget op, for spillet var konstrueret med det formål at underholde og ikke for at foretage en systematisk videnskabelig undersøgelse.

Jeg stiftede første gang bekendtskab med LEGO Spybotics ved en præsentation i foråret 2002, nogle måneder før lanceringen af produktet. De små robotter havde et mere råt samtidigt mere stilfuldt design end tidligere robotprodukter fra LEGO og de fleste andre robotprodukter for den sags skyld. Figur 1.2 viser en Spybot m. fjernbetjening. Der er fire forskellige robotter; alle med et futuristik udseende, hvor de klassiske grå LEGO klodser i LEGO Spybotics produktet er udskiftet med klodser med metallic look og alle er udstyret med en "laserkanon".

FIGUR 1.2: Gigsmesh G60, en af de fire forskellige LEGO Spybots, m. fjernbetjening.

Jeg fik sidenhen rig lejlighed til at stifte nærmere bekendtskab med dem og fik endvidere mulighed for at opleve, hvordan både børn og voksne oplever Spybotics produktet. Det skete ved at konstruere et robotspil i samarbejde med Christian. Robotspillet blev afprøvet ved kulturnatten i Århus i oktober 2002, ved NordiCHI konferencen oktober 2002[1] og på Huset[2] Århus i december 2002. Jeg oplevede ligeledes, hvordan børn kan lege med Spybots, da jeg var "robotkyndig" i forbindelse med en emneuge i børnehaveklassen 0.B på Katrinebjergskolen, hvor robotter var temaet. Spilscenarierne og oplevelserne i 0.B er beskrevet yderligere i kapitel 5.

Med de baggrund i de praktiske erfaringer med Spybotics oprettede Christian undervisningsforløbet *Computerspil som æstetik/LEGO Spybotics*. Jeg var instruktor på kurset og varetog undervisningen i forbindelse med kunstig intelligens i computerspil. Kurset bestod af en teoretisk del samt en praktisk del hvor LEGO Spybotics og traditionelle computerspil indgik. Pointen med at benytte Spybots i et kursus om computerspil som æstetik, var at give de studerende mulighed for at opbygge spil på kort tid uden nødvendigvis at have programmeringsmæssige forudsætninger.

[1] Nordisk konference for menneske-maskine-interaktion (HCI) [10]
[2] Huset er en kultur- og aktivitetsinstitution midt i Århus

En af konklusionerne, vi nåede frem til i løbet af kurset, var, at Spybots i sig selv har et tiltalende design og er morsom som fjernstyret bil, men at de medfølgende spil er for simple, og at koblingen mellem den fiktive verden, der udfolder sig på skærmen og det spil, der foregår i det fysiske rum ikke hænger sammen. "Gør det selv spil" delen, hvor spilleren kan konstruere sine egne missioner, er ganske interessant og mægtig underholdende, men spillerens mulighed for at programmere sine Spybots er alt for simpel.

De fysiske spil med Spybots som henholdsvis aktant, styret af en spiller og som autonome modstandere overbeviste mig om, at robotspil kan være en både underholdende og udfordrende aktivitet.

1.2 Motivation

Erfaringerne med LEGO Spybotics er inspirationskilde til dette projekt. Efter afprøvningen af de computerspilsinspirerede robotspil, var jeg interesseret i at undersøge om et robotspil kunne konstrueres endnu bedre. Adfærden for modstanderrobottene i de tidlige eksperimenter var nemlig alt for ofte irrationel grænsende til tåbelig; de kendte blandt andet ikke forskel på "ven" og "fjende". Derfor er der en datalogisk udfordring i at skabe modstanderrobotter, hvis adfærd er fornuftig og på den måde højne robotspillets kvalitet. Det ville specielt være en udfordrende faktor, hvis robotmodstanderne i et spil skal være i stand til at tilpasse deres strategi til spillerens strategi og dermed blive bedre modstandere. Dette projekt handler om, hvordan robotspil vil kunne realiseres. Emnevalget er udsprunget af tre årsager:

1. Interesse for at arbejde med robotter.

2. Lysten og udfordringen ved at arbejde med børn.

3. En stor passion for computerspil.

I løbet af min uddannelse har jeg stiftet bekendskab med robotter i forbindelse med overbygningskurserne *Adaptive robots* og *Embedded Systems Embodied Agents*, som jeg fulgte i 2002 og 2003. Gennem disse kurser fik jeg øjnene op for de mange anvendelsesmuligheder, der er for robotter, men også hvor mange udfordringer der er ved at programmere dem.

Lysten til at arbejdet med børn, computere og programmering stammer delvist fra det teoretisk fundament, jeg fik opbygget gennem det datalogiske overbygningskursus *Children as Programmers*. Men lysten stammer nok i endnu højere grad fra de projekter, hvor jeg har arbejdet med børn, computere og programmering blandt andet et programmeringsprojekt i forbindelse med *Dansk naturvidenskabsfestival 2002*, hvor jeg var en del af en gruppe, der lærte gymnasieelever at programmere robotter. Jeg har været en del af et projekt i en børnehaveklasse,

hvor eleverne skulle bygge og arbejde med robotter i 2003. Desuden var jeg robottræner i forbindelse med *First LEGO League 2004*; en robotkonkurrence hvor børn under 16 år fra hele Norden skulle programmere robotter til at løse en række forskellige opgaver [3].

Interessen for computerspil stammer tilbage fra midten af 80'erne, men der blev skabt et fagligt fundament i forbindelse med kurset *Computerspil som Æstetik/LEGO Spybotics*, hvor undervisningen blev varetaget af Christian, og gennem læsning af bøger og artikler om computerspil med hovedvægt på den tekniske del.

Disse tre interesser forenes ved arbejdet med udvikling af et robotspil til børn. Programmer til robotter skal udvikles og implementeres. Spillet skal udvikles og afprøves i samarbejde med børn. Computerspil er inspirationskilde og arbejdsredskab.

1.3 Mål

Målet er at implementere et robotspil for børn i aldersgruppen 9-13 år, der er den målgruppe som Spybotics produktet henvender sig til, med Spybots som aktører. Spybots er valgt dels fordi jeg på baggrund af allerede høstede erfaringer ved, at de kan anvendes til formålet og dels fordi de er tilgængelige for mig. I løbet af processen er der en række spørgsmål, der er interessante og nødvendige at få belyst for at opnå bedre autonome modstanderrobotter i robotspil og dermed bedre robotspil i det hele taget. Målet med projektet er at undersøge sådanne spørgsmål:

- Hvordan kan et sjovt og udfordrende robotspil for børn i alderen 9-13 udformes ?
- Er det muligt og fornuftigt at overføre elementer fra computerspil til robotspil ?
- Hvilken type kontrolprogram skal en agent, der er modstanderen i et robotspil have og bør den være adaptiv ?
- Hvilke robotprodukter er velegnede til at indgå i et robotspil, når der skal tages hensyn til både tekniske egenskaber og prisen ?

Projektet med at skabe et robotspil kan ses som et forstudie til en produktion af robotspil. Målgruppen for robotspillet vil være børn i alderen 9-13 år som Spybotics produktet. Derfor vil økonomien i forbindelse med konstruktion af robotspil også spille en rolle.

1.4 Metoder og værktøj

Dette projekt er nok lidt atypisk fordi den datalogiske undersøgelse ikke er projektets mål. Datalogien benyttes som et middel til at nå et mål som er et godt

robotspil. Dermed breder jeg mig ud over de rammer der traditionelt anses for at være datalogi.

For at finde frem til hvordan et godt robotspil kan være udformet, vil computerspil blive undersøgt, idet de er meget udbredte og populære. Da både computerspil og robotspil er interaktive og involverer brugen af computere, er det nærliggende at antage, at elementer fra computerspil kan overføres til robotspil. Computerspil bliver gjort til genstand for undersøgelse for at finde paralleller mellem robotspil og computerspil. Computerspil er opbygget af tekniske og æstetiske elementer. Jeg har medtaget den æstetiske dimension, fordi den er vigtig at holde sig for øje når den tekniske del udformes. Mit fokus er på den tekniske udvikling. På grund af tids- og pladshensyn har det ikke været muligt at udfolde det æstetiske element. I en efterfølgende produktion af robotspil vil det være en dimension der skal lægges større vægt på.

Med baggrund i hvordan agenter benyttes i computerspil, vil agenter generelt blive undersøgt for at afdække, hvordan et kontrolprogram til en robot kan udformes, hvis den succesfuldt skal indgå som autonom modstander i et robotspil. På trods af at Spybotten allerede er blevet valgt som den tekniske platform, vil en række kommercielle robotprodukter blive gennemgået, med henblik på at finde en eller flere, der ville være egnede til formålet, hvis jeg ikke var bundet af økonomiske begrænsninger. Herefter kan kontrolprogrammet kan implementeres.

Når et kontrolprogram er blevet implementeret og overført til den valgte robot, bliver programmet afprøvet på to måder. Dels ved at lade to grupper af adaptive autonome robotmodstandere kæmpe imod hinanden og dels ved at lade robotmodstanderen indgå i robotspil, som vil blive afprøvet sammen med nogle børn i målgruppen.

1.5 Specialets struktur

Specialet falder naturligt i tre dele. Første del er en teoretisk oversigt omkring spil generelt og computerspil i særdeleshed. Anden del er en teoretisk dybdegående gennemgang af agenter, hvor et konkret robotprodukt vælges til at indgå i et robotspil. Tredje del består af en nærmere undersøgelse af Spybots samt implementering og afprøvning af et agentmodstanderprogram på Spybotten. Nedenfor er en oversigt over indholdet i specialets kapitler.

Kapitel 2 indeholder et oprids af forskellige spiltyper for at placere robotspil i forhold til andre typer spil.

Kapitel 3 gennemgår forskellige tekniske og æstetiske elementer, som computerspil indeholder, samt et oprids af forskellig computerspilgenrer.

Kapitel 4 indeholder definitioner af agenter og desuden generel gennemgang af agenter og agentbegrebet.

Kapitel 5 ser nærmere på robotter og kommunikation samt forskellige robotprodukter med henblik på at undersøge hvilke robotter, der er velegnede til at indgå i en prototype af et robotspil.

Kapitel 6 indeholder en undersøgelse af Spybotten, der er den hardwareplatform, som skal benyttes i forbindelse med prototypen til et robotspil.
Kapitel 7 indeholder en gennemgang af Spybottens arkitektur samt arkitekturen af det kontrolprogram, der skal implementeres i Spybot modstanderne.
Kapitel 8 indeholder afprøvningerne af kontrolprogrammet, der blev implementeret i Spybot modstanderne, der indgår i spillet, samt resultaterne fra afprøvningerne.
Kapitel 9 indeholder specialets konklusion samt en perspektivering, der forklarer, hvordan der kunne arbejdes med udviklingen af robotspil, hvis der havde været mere tid til rådighed.
Appendix A indeholder en kort beskrivelse af en række computerspil, som der refereres til rundt omkring i specialet.
Appendix B indeholder en CDROM med programmer, billeder og dokumentation, der er relevant i forhold til specialet.

I løbet af specialet vil vigtige begreber, der bliver introduceret blive markeret med **fed** og *kursiv* vil blive benyttet til at markere titler og navne som fx *Embodied Evolution* og *Spybot Engine*.

Del I
Hvad er computerspil

Indtil nu ved vi, at robotspil i rummet kan være underholdende og udfordrende. Samtidigt er det klart, at der er nogle problemer ved de robotspil som hører sammen med LEGO Spybotics produktet. Der er ligeledes problemer i forbindelse med robotspil, hvor modstanderrobotternes kontrolprogrammer er implementeret ved hjælp af LEGO Spybotics "gør det selv" programmeringsdel.

Har man intentioner om at konstruere et fornuftigt robotspil, vil en analyse af eksisterende spil fra computerspil til *Kalaha*, med henblik på at kunne forbedre robotspil, givet vist være et ganske udemærket udgangspunkt. Det er forhåbningen at der vil kunne lånes elementer fra nogle spiltyper og overføre dem til til robotspil.

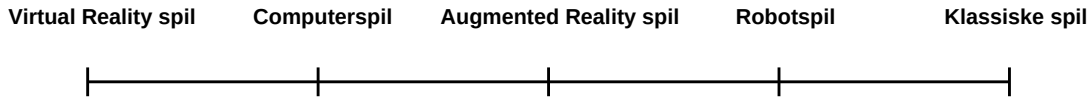
Derfor vil jeg i denne del af specialet først se på spil generelt og herefter se nærmere på computerspil. Computerspil består af to grundpiller: de æstetiske elementer og de tekniske elementer. Disse grundpiller vil blive gennemgået for at afdække, hvilke dele som med fordel kan benyttes i robotspil.

2 Spiltyper

Mennesket har sandsynligvis leget og spillet, så længe vi har eksisteret. Af eksempler på gamle spil, der stadig spilles i dag kan nævnes det over 7000 år gamle Ægyptiske spil *Kalaha*. Spils indhold og fremtoning har gennemgået en rivende udvikling siden da; især efter at de første computerspil opstod i 1950'erne.

Det skal bemærkes at distinktionen mellem spil og leg er noget særegent for de germanske sprog. Når spil eller leg benyttes, menes der med *spil* en legende aktivitet der i højere grad end andre typer leg benytter nedskrevne regler. Ligeledes er spil konkurrenceprægende og har ofte en vinder. I leg opstår reglerne mere spontant og kan ændres under forløbet. Legen er sjældent konkurencepræget. Det skal bemærkes at grænsen mellem spil og leg er udflydende [1].

Når der tales om spil er det vigtigt at gøre sig klart hvilken type spil der er tale om. Figur 2.1 illustrerer hvor stor del af spillets univers, det rum spillet udspænder, der er opbygget af computere. Virtual Reality (VR) spil længst til venstre er de mest digitale/elektroniske, hvorimod de klassiske spil ikke indeholder digitale elementer overhovedet og består af kortspil, brætspil, terningspil etc. *Klassiske spil* er måske en lidt inkonsekvent betegnelse, da mange populære brætspil, som Trivial Persuit, er nyere end de gamle computerspil. Men da selv de nyere spil inden for kategorien *klassiske spil* spilles efter samme principper som gamle spil som kortspil, skak og ludo, i form af fysisk tilstedeværelse på samme sted, direkte manipulation af spillets elementer, brikker, kort og terninger; samt ingen involvering af elektroniske komponenter, benyttes betegnelsen *klassiske spil* alligevel.



FIGUR 2.1: Oversigt over forskellige spiltyper med større eller mindre grad af computerelementer

Nedenfor følger en kort beskrivelse af de fem spiltyper, som er illustreret i figur 2.1. Beskrivelserne skal danne et indtryk af, hvilke områder der findes spil inden for, samt hvilke typer spil der er udbredt og hvilke der ikke er.

Virtual Reality spil Disse spil foregår udelukkende i den virtuelle verden. VR spil implicerer ofte en hjelm med display til øjnene og handsker eller endda til tider heldragter udstyret med elektroder. Den omfattende og dyre udrustning gør spilleren i stand til at forsvinde fuldstændigt ind i en kunstigt opbygget verden, hvor alt inklusiv spilleren selv er virtuelt. VR hjelmen gør, at uanset hvor spilleren ser hen, vil han kun se den virtuelle verden

[1] For en grundigere diskussion af forholdet mellem spil og leg, henvises til [37] kapitel 4

og handskerne gør ham i stand at interagere med objekter i den virtuelle verden.

Computerspil I computerspil foregår spillet på en skærm foran spilleren og spilleren kan kun indirekte påvirke spiluniverset ved hjælp af tastatur, mus, joystik, gamepad, dansetæpper etc. I praksis er det min erfaring at spilleren har lige så stor mulighed for at bliver opslugt af det univers, spillet udspænder, som i VR spil. Det påkrævede udstyr for at kunne spille er langt billigere end for VR og blandt andet derfor er computerspil langt mere udbredte end VR spil. Der findes en lang række kommercielle spilprodukter inden for dette områder og disse kan afvikles på en bred vifte af tekniske platforme, spændende fra mobiltelefoner over PC'er til spillekonsoller tilsluttet et fjernsyn. Af kommercielle produkter kan nævnes *Pong, Pacman, Tetris, Wolfenstein 3D, Splinter Cell* og et utal af andre spil. Se appendix A for beskrivelse.

Computerspil vil blive gennemgået i større detalje i næste afsnit da denne type spil har mange elementer til fælles med de interaktive robotspil i det fysiske rum. Samtidigt er computerspillenes store popularitet endnu en grund til at se nærmere på dem, idet de populære aspekter af computerspil eventuelt vil kunne overføres til interaktive robotspil i det fysiske rum.

Augmented Reality spil Inden for forskningsområdet augmented reality (AR) [16, 17, 43], der betyder beriget virkelighed, forsøger man at lægge et digitalt eller virtuelt lag oven på den virkelige verden. Denne teknologi gør fx arkitekter i stand til at placere en virtuel bygning på en byggegrund for at give et overblik over, hvordan bygningen kommer til at tage sig ud i de virkelige fysiske omgivelser. Spil inden for denne genre forsøger at blande de bedste elementer fra den virtuelle og den fysiske verden. Der er endnu ingen kommercielle AR spilprodukter, men af lovende projekter kan blandt andet nævnes det augmenterede brætspil BattleBoard 3D [59]. I BattleBoard 3D sidder to spillere overfor hinanden med en spilleplade imellem sig som ved andre topersoners brætspil. På pladen er der spillebrikker som ved mange andre brætspil. For at kunne spille spillet skal hver spiller have et par briller på forsynet med et kamera foran øjnene og et display til hvert øje. Med brillerne på kan spilleren se en lille animeret kriger oven på hver af de fysiske brikker. Hvis en spiller ønsker at indtage et felt, hvorpå der står en anden brik, så udkæmper de små en fysisk kamp på liv og død. Spilleren til den faldne kriger må herpå flytte sin brik fra spillepladen.

Robotspil Robotspil befinder sig på det stadie som computerspillet befandt sig på for 30-40 år siden - for at sige det som det er, spilles denne type spil næsten udelukkende af entusiaster) inden for universitetsverdenen. Der findes en række spil inden for området, mest kendt, inden for disse kredse, er

robotfodbold. Som navnet antyder foregår spillet mellem to hold på en bane med et mål i hver ende og formålet med spillet er at få bolden så mange gange i modstanderens mål som muligt [11].

Første og eneste kommercielle produkt inden for denne kategori er, til mit kendskab, LEGO Spybotics produktet.

Klassiske spil De klassiske spil, brætspil, terningspil og kortspil, behøver ikke den store præsentation og er stadig de mest udbredte af de fem kategorier, selv om de ikke nødvendigvis de de mest spillede af personer i alle aldersgrupper. Af brætspil kan nævnes *Skak, RISK, Backgammon, Matador* og tusindvis af andre. Udbredte terningspil er spil som *Yatzi* og *Mayer* og af kortspil kan nævnes *Whisth, Bridge, Poker og 500*.

Den største forskel på robotspil og de tre førstnævnte spilkategorier er at robotspil udelukkende foregår i den virkelige verden. Det eneste digitale element i spillene er den computer, der befinder sig inden i robotterne.

Det er muligheden for at konstruere spil inden for robotspilkategorien, der vil blive fokuceret på i dette speciale. Der vil også blive fokuceret på computerspil for at lære af de elementer der ligger til grund for computerspillenes succes og for at kunne forbedre robotspil ved at overføre elementer fra computerspil til robotspil.

3 Computerspil

I dette afsnit vil jeg fremdrage en række karakteristika ved almindelige computerspil af både teknisk og æstetisk karakter. Formålet er at finde frem til centrale elementer ved computerspil, som kan overføres til interaktive robotspil. Når begrebet "interantiv" benyttes om spil menes der et spil, hvor spilleren kan gribe ind i spillets forløb. Robotfodbold er ikke interaktivt, for når spillet er startet kan spillerne blot læne sig tilbage og overvære spillets forløb. *Skak* er derimod et interaktivt spil. I det følgende vil der blive refereret til en række forskellige computerspil og en beskrivelse af disse findes i appendix A side 119.

Som det blev nævnt i indledningen, så er computerspil særdeles populære og meget udbredte. Da der samtidigt er tale om interaktive spil, er det oplagt at analysere disse med henblik på at overføre elementer fra virtuelle interaktive spil, til de interaktive robotspil som jeg ønsker at konstruere.

Computerspil er dragende. Siden deres første store kommercielle succes PONG, fra spilproducentern ATARI i 1972, er udbredelsen af spil steget stødt år efter år [35] . I det følgende vil jeg opridse og forklare nogle af de mest centrale **æstetiske elementer**, der kendetegner computerspil. Herpå vil der følge et oprids af de vigtigste **tekniske elementer**, som et computerspil indeholder.

Computerspil kan på mange måder siges at bygge bro mellem de klassiske spil og den fiktive verden i tegneserier, film og bøger. Fra den fiktive verden har computerspillet taget det narrative element eller **fortællingen**.

Fra de klassiske spil har computerspillet lånt det interaktive element; der gør, at det rent faktisk er muligt at påvirke spillets forløb. Inden for computerspilsterminologien kaldes dette element **gameplay** [53]. Gameplay vil blive diskuteret nedenfor. Disse elementer vil blive betegnet som æstetiske.

De tekniske elementer er som navnet siger teknisk betonet, hvilket i denne sammenhæng vil sige, at det er direkte forbundet med spillets programmeringskode eller den hardwareplatform, som spillet afvikles på. De tekniske elementer omfatter **emergens**, der er et begreb der benyttes når et systems helhed er mere end summen af de enkeltdele som det består af. Et andet teknisk element er den **kunstige intelligens** (KI) i spillene, der er en betegnelse for de processer der ligger til grund for en agents adfærd.

Distinktionen mellem det tekniske og det æstetiske kan virke som en smule kunstig, fordi de er to sider af samme sag. Når opdelingen alligevel er der tale om et analytisk greb der er foretaget for at sætte fokus på de æstetiske og tekniske elementer hver for sig.

Inden for computerspillets verden er der et nært sammenspil mellem de tekniske og de æstetiske elementer. Tager man som eksempel et spil som *Splinter Cell*, der er et skydespil, vil et dårligt implementeret teknisk element som den kunstige intelligens have stor betydning for den æstetiske oplevelse. Kommer aktanten ind i en rum fyldt med terrorister, og disse dels begynder at skyde på hinanden og møblementet i rummet og dels løbe forvirrede rundt, vil det bryde illusionen om,

at spilleren kæmper imod en toptrænet hær af professionelle dræbere.

Når opdelingen mellem teknisk og æstetisk er valgt og begge dele behandles og anses for vigtige i et datalogisk speciale, skyldes det at disse dele hænger uadskilleligt sammen i computerspil. I forbindelse med implementering af computerspil er det vigtigt at være pinligt bevist om at enhver teknisk designovervejelse kan have betydning for den æstetiske oplevelse af spillet og dermed om spillet overhovedet bliver værd at spille. Omvendt har den æstetiske side stor betydning i valg af teknologisk design.

Efter gennemgangen af de tekniske og æstetiske elementer vil en række **computerspilsgenrer** blive opstillet, så det bliver muligt af afgøre, hvilken type robotspil, der senere skal implementeres.

3.1 Æstetiske elementer

Et væsentligt element som computerspillet har lånt fra fiktionen er det narrative- eller fortællende element. **Fortællingen** opbygger et fiktivt univers omkring computerspillet både før og under selve spillet. Terminologien omkring fortællingen i spil er hentet fra [53] kap. 12.

Et andet centralt element er belønningsprincippet, der kan forekomme i flere forskellige former. **Point**, der tildeles, når spilleren handler godt, på samme måde som det kendes fra spil som dart. Belønningen kan også være at få adgang til et nyt **level** og sidst men ikke mindst kan belønningen være nyt udstyr eller forøget styrke, der gør det lettere at komme videre i spillet [53]. Sidst men ikke mindst er der gameplay.

3.1.1 Fortællingen

Fortælling er en af de væsentligste forskelle mellem de klassiske spil og computerspil. Fortællingen kan udspænde sig; både før spillet som en form for forhistorie eller under selve spillet. Man skelner mellem to fortælleformer; den første er den traditionelle som kaldes out-of-game, hvor selve spillet indstilles og spilleren introduceres for nye udfordringer i form af tekst, lyd, animerede sekvenser eller en blanding af disse tre. Den anden fortælleform, som kaldes in-game fortælling lader historien udfolde sig, imens spilleren er i gang med spillet.

Forløbet af historien i spillet har ofte en begyndelse, en midte og en afslutning. Afslutningen vil ofte indtræffe, når spilleren har opnået sit mål eller er blevet sat ud af spillet. I *Doom* bliver spilleren briefet om, at væsner fra helvede er begyndt at dukke op og de skal bekæmpes. Spillet afsluttes, når spilleren overvinder alle monstre eller at de fæle monstre får bugt med spilleren. Undervejs i hele spillet er der delmål, der er en central del af belønningen i computerspil.

Det fortællende element får en mere og mere central rolle i computerspillene. Eksempler på sådanne spil er actionspillet *Splinter Cell*, hvor spilleren er den

hemmelig agent Sam Fisher, der skal bekæmpe den onde terrorist Vlademir Nicholiev og hans organisation samt uskadeligegøre det ultimative terrorvåben *The Ark*. Fortællingen udfoldes som en forhistorie før hver bane, altså en out-of-game fortælling. Dele af historien fortælles under spillet, hvor der gives intrukser via. en PDA som Fisher medbringer; der opsamles oplysninger fra computere, faldne modstandere og der er desuden mulighed for at samtale med og afhøre agenter. Her er der tale om in-game fortælling.

3.1.2 Belønning

Den mest benyttede form for belønning er points. Spillerens pointantal forøges, når denne gør noget godt, hvilket opildner ham til at fortsætte på samme måde. Point giver spilleren mulighed for at sammenligne hans præstation med andre spilleres og dermed det Richard Rouse betegner *bragging rights*([53], som er retten til at prale, hvis man er bedre til et givet spil, end andre er. Desuden giver point spilleren mulighed for at måle sin egen præstation i forhold til sine tidligere spil.

En anden udbredt form for belønning i computerspil er levels. Levels er med til at holde spilleren fanget fordi denne ofte er nysgerrig efter at se, hvad næste bane byder på. Næsten alle spil benytter sig af levels i en eller anden form. I *Pacman, Wolfenstein 3D* og mange andre starter man på en helt ny bane, efter en bane er gennemført. *Civilazation II* har en anden form for levels, når en spiller gør opdagelse som hjulet eller smedning får denne muligher for at gøre nye ting, bygge bedre bygninger og krigere. Dermed kommer der en udvikling i spillet, der svarer til at begynde på et nyt level.

Der findes dog også spil, hvor belønningen gives mere inddirekte et eksempel herpå er opbygningsspil som Sims City, hvor spilleren skal opbygge en by med alt hvad der hører til af institutioner, infrastruktur og meget andet. Belønningen her består kort og godt i glæden ved at opbygge et velfungerende samfund.

3.1.3 Gameplay

Gamplay er betegnelsen for computerspillets interaktivitet. Spildesigneren Richard Rouse definerer gameplay således:

> The gameplay is the component of computergames which is found in no other art form: interactivity. A game's gameplay is the degree and nature of the interactivity that the game includes, i.e., how the player is abel to interact with the game-world and how the game-world react to the choices the player makes. [53] [introduction xviii]

I Space Invaders er gameplayet at bevæge sit rumskib frem og tilbage i bunden af skærmen og skyde fjendtlige rumskibe. I Civilization II er gameplayet at opbygge et samfund, udforske verden, opbygge hærenheder, gøre opdagelser. Gameplayet

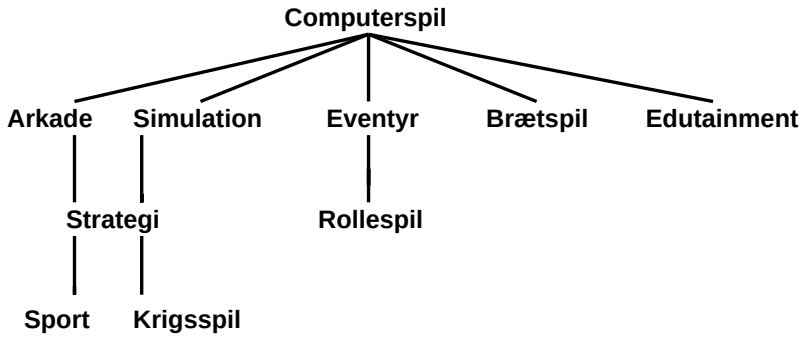
i Splinter Cell er at bekæmpe terror ved at overvåge, aflytte og nedkæmpe terrorister.

Gameplayet hænger nøje sammen med, hvilken teknologi der stilles til rådighed, graden af interaktivitet og måden der interageres med spillet på er forskellig alt efter om spilleren er udstyret med et tastatur, en gamepad en mul eller en lyspistolsamt. Gameplayet hænger også sammen med den genre spillet ligger inden for. I et rollespil hvor udforskning er et vigtigt element bør spilleren have mulighed for at navigere rundt næsten ubegrænset, men i et arkadespil som *Tetris* har spilleren udelukkende mulighed for at flytte på de faldende figure.

3.2 Computerspilgenrer

Inden konstruktionen af et nyt spil kan det være nyttigt at gøre sig nogle tanker om hviken genre spillet skal tilhøre. Genrevalget har i fiktionens verden stor indflydelse på den målgruppe det er muligt at nå og hvilket budskab forfatteren ønsker at aflevere. Forskellige genrer af computerspil kræver også forskellige forudsætninger fra spilleren. Nogle spil stiller krav om strategisk tænkning og overblik og spil med de karakteristika vil eksempelvist sjældent være velegnede til mindre børn.

Lars Konzack opstiller ni forskellige forskellige computerspilgenrer [37]. Disse er gengivet i figur 3.1. Nedenfor er de ni forskellige spilgenre kort beskrevet.



FIGUR 3.1: Oversigt over computerspilgenrer [37]

Arkade Arkadespillene er kendetegnet ved færdighed og præstation. Arkade er en historisk funderet betegnelse der stammer fra 70'ernes spillearkader; i dag benyttes betegnelsen **actionspil** oftere[1]. Eksempler på arkadespil er *Tetris, Pacman, Spaceinvaders, Doom* og *Quake*. Disse spil er ofte lette at komme i gang med, men det kræver øvelse at blive en dygtig spiller.

[1]Konzak mener, at arkade er en mere præcis betegnelse end actionspil da actionspil, ofte benyttes om tempofyldte spil uanset genre (Kilde: Lars Konzak selv)

Strategi Strategispil kræver overblik, logisk tænkning, analyse og vurdering. De er ofte inspireret af historiske begivenheder eller perioder. Her er tale om spil som det klassiske *Minestryger* og *Civilazation II*. Inden for disse spil er læsning af manualer enten i bogform eller på skærmen ofte nødvendigt for at opnå et tilfredsstillende resultat.

Simulation Drejer sig om simulation af blandt andet fly og biler. I denne genre lægges der vægt på, at efterligne virkeligheden i så høj grad som muligt. Visse simulationsspil bliver udviklet direkte med henblik på at uddanne piloter og lastbilschauffører.

Sportsspil Disse spil minder meget om arkadespillene; blot er temaet sport. Diverse racerbilspil, ishockeyspil og fodboldspil. Sportsmanagerspil falder inden for strategispil.

Krigsspil Krigsspil er simulationsspil, ofte med arkadeelementer, der har krig som tema. *StarCraft* og spil hvor verdensherredømmet er spillets mål, er eksempel på et sådanne spil.

Eventyr I eventyrspillene, også ofte kaldet adventurespil, handler det om at udforske og opleve, så det gælder om at være nysgerrig, tålmodig og have god iagttagelsesevne.

Rollespil Er en udvidelse af eventyrspil. Her trækkes på traditionen fra de fysiske rollespil som *Dungeons & Dragons*, hvor der stilles store krav til indlevelse i det fiktive univers.

Kort- og brætspil Disse spil er computerspilsudgaver af de spil der i figur 2.1 side 9, blev refereret til som klassiske spil. Eksempler på denne kategori er *Skak, Syvkabale* og *fire på stribe*.

Edutainment Spil, hvor man forsøger at få læring og underholdning til at gå op i en højere enhed. Et eksempl er de danske *Pixiline* spil.

Der findes også en lang række spil spil der er hybrider mellem flere genrer. Spil som *Tomb Raider*, der i hovedrollen har Laura Croft, den moderne kvindelige pendent til Indiana Jones, er en blanding mellem arkadespillenes action- og adventurespillenes opdagelseselement.

Der er en række spil, der er vanskeligeligt at implementere i praksis, hvis man ønsker at skabe et robotspil. Det er eksempeltvist vanskeligt at skabe et simulationsspil, da en simulation er en efteligning af noget andet og det er vanskeligt at få en robot til at efterligne andet end en robot. Tidligere blev robotfodbold nævnt og det kunne jo umiddelbart lyde som en form for simulation - og det bliver robotfodbold muligvis med tiden[2], men med simulation forstås en gengivelse

[2]Arrangørene mener selv, at den første fodboldkamp mellem et hold af menneskelige- og robotspillere vil finde sted senest i år 2050 [11]

af et hændelsesforløb fra den virkelige verden [18] og i robotfodbold er reglerne modificeret til næsten ugenkendelighed. Derfor er der snarere tale om en blanding mellem strategi og edutainment, da det kræver overblik, logisk tænkning og analyse at skrive et kontrolprogram til en robot, samtidigt med at programmøren lærer noget nyt ved at skrive et program til en robot.

Et underholdende spil baseret på mindre robotter som eksempelvist Spybots burde på baggrund af de praktiske erfaringer fra kulturnat og børnehaveklasse være et spil der er let at komme i gang med, men samtidigt et spil, der godt må kræve noget af spilleren. Derfor vil en form for arkadespil være et fornuftigt genrevalg.

Spillet kunne dog nok tilføjes en dimension, hvis der blev tilføjet en rammefortælling. In-game elementer, hvor spilrobotten gav spilleren instruktioner undervejs, ville ligeledes gøre spillet mere interessant.

3.3 Tekniske elementer

De tekniske elementer er direkte relateret til hardware, programmeringskode eller begge dele. Et teknisk element kan være, at spilleren skal have mulighed for at kontrollere den aktant, som agerer i spilverdenen. Så hvis man spiller *Pacman* skal der være mulighed for at styre den lille ost[3], der er hovedperson. I praksis begrænses et spils tekniske elementer af, hvad der er teknisk muligt. Da der er tekniske begrænsninger for spils grafik, lyd og gameplay er det spildesignerens opgave at konstruere et så godt og underholdende spil som det kan lade sig gøre inden for de tekniske rammer, der kan lade sig gøre [53].

Et spil skal indeholde en aktant eller et element, som brugeren af spillet skal kunne kontrollere, da der ellers blot ville være tale om en demonstration eller en film. Et eksempel på en aktant i et spil er osten i Pacman spillet.

Det bringer os frem til andet punkt på listen af tekniske elementer, som et spil bør indeholde; nemlig modstandere af en eller anden form. Modstandere skal forstås meget bredt og dækker over alt lige fra spøgelserne i *Pacman* til de faldende brikker i *Tetris*. Ligeledes kan modstanderen være en del af omgivelserne; fx en dør der åbner og lukker med et bestemt interval. Modstanderen eller modstanderne er ofte computerstyrede, men kan også være styret af andre personer.

Sidst men ikke mindst skal der være en form for omgivelser, som aktørerne bevæger sig rundt i. Det kan være hele verden som i *Civilazation II*, skumle 3D labyrinter, som i *Doom* eller en 2D labyrint som i *Pacman*.

[3]Faktisk er der ikke tale om en ost men en hockeypuck, men da spillet skulle stå i arkader var producenterne bange for, at spillerene skulle ridse midten af P'et væk, så der kom til at stå Fuckman. Derfor blev navnet ændret til *Pacman* [55].

3.3.1 Emergens

For at et spil kan være udfordrende er det vigtigt, at der er overraskende elementer i det. I bogen *EMERGENCE From Chaos to Order* skriver John Holland blandt andet om emergens i blandt andet brætspil[4] [31]. Desuden skriver han om hvordan logiske systemer som computere kan opnå kompleks og uforudsigelig "opførsel". Det er her, at emergens kommer ind i billedet. Ordet emergens kommer fra latin og betyder "dukke op". Holland benytter begrebet til at betegne det fænomen, når et systems helhed er mere end summen af dens dele. Der er med andre ord tale om, at der opstår noget uforudset i et logisk, og ofte enkelt system. Holland benytter blandt andet begrebet i forbindelse med brætspil som kryds og bolle og skak, hvor der i sidstnævnte tilfælde opstår et uhyre komplekst spil på baggrund af simple regler.

Han benytter betegnelsen *tilstand* om et spils konfiguration, det vil sige placering af brikker på brædtet på et givet tidspunkt. Hver gang en spiller flytter en brik ændres spillets tilstand. Et spil går i bund og grund ud på at opnå en lovlig tilstand, hvor man har vundet. For at det kan opnås, må man have en strategi, og det er her muligheden for udfordring i et spil opstår. Er det muligt at gennemskue modstanderens strategi bliver spillet forudsigeligt og dermed kedligt. Af samme grund betragtes kryds og bolle som et børnespil, for næsten alle voksne vil kunne forudsige sin modspillers træk. I skak er det derimod yderst kompliceret at forudsige hvad en modspiller vil foretage sig de næste par træk, på trods af at reglerne for skak er forholdsvis enkle.

På samme måde forholder det sig med computerspil. Er man i stand til at gennemskue spillets logik alt for hurtigt er det som udgangspunkt ikke sjovt at spille særligt længe. For en del spils vedkommende består en del af udfordringen i at gennemskue logikken bag ved spillet. Et computerspil som det klassiske *Tetris*, er et godt eksempel på et uhyre simpelt spil, som på grund af den høje grad af emergens alligevel er sjovt at spille. Spilleren ved, at der vil falde en af de seks typer klodser ned, samt at hver klods fylder fire felter. Alligevel er spillet udfordrende, for selv om spilleren stort set ved, hvad der venter ham, er det i længden svært at få de faldende klodser til at passe sammen. Og når tempoet på de faldende klodser øges gradvist stiger sværhedsgraden ligeledes.

3.3.2 Kunstig intelligens

Kunstig intelligens (KI) blev oprindeligt introduceret for at vi mennesker kunne få indsigt i, hvordan vores hjerner fungerer. Traditionelt set har det altså været KI forskernes mission at udforme programmer, der fungerede som os. Af samme grund formulerede Alan Turing i 1950 en test for kunstig intelligens således; hvis en person sættes foran en terminal, hvor han skal kommunikere tekstuelt med

[4]Faktisk er Hollands ærinde at lade læseren få indblik i, hvordan man ved hjælp af computer kan forstå komplekse systemer som eksempelvis vejret

en person eller den KI der skal testes. Kan testpersonen ikke afgøre, om han har kommunikeret med en KI eller en person og han har kommunikeret med KI'en, så har KI'en bestået Turing testen [54].

Der findes mange forskellige definitioner på hvad KI egentligt er. Når vi beskæftiger os med KI inden for computerspil, betyder det imidlertid ikke noget om computeren tænker eller opfører sig som et menneske. Spillet skal spilles af en menneskelig spiller, og så længe spillerens modstander opfører sig fornuftigt, er det ligegyldigt, om modstanderen er intelligent eller ikke. Kunstig fornuft ville derfor være en mere passende betegnelse i forbindelse med spil, men det er nu en gang konvention at benytte begrebet kunstig intelligens, så det vil jeg også gøre. Med andre ord vil KI i det følgende blive benyttet om den kode der styrer spillets aktanter, hvilket er samme definition som Rouse benytter [53]. Som en krølle på halen skal det nævnes, at nogle forskere inden for kunstig intelligens lader sig inspirere af de interaktive underholdningsspil [56].

KI i computerspil kan være alt fra en skaksimulator som Deep Blue, skaksimulatoren der vandt over Kasparov, eller algoritmen der ligger til grund for opførslen for de monstre, nazister og andre kryb, man støder på i arkadespil som *Doom* eller *Wolfenstein 3D*. Der er dog stor forskel på den måde, de to ovennævnte KI'er er implementeret på. Deep Blue skal kun operere inden for en simpel og fuldstændig kendt verden, bestående af brikker og brædt. Verden er statisk, imens næste træk udtænkes og det betyder mindre om et træk tager et par nanosekunder eller et par minutter. Denne type KI kaldes også **GOFAI (Good Old Fasion Artificial Intelligence)**. Strategien for en GOFAI er lidt simplificeret at skabe et billede af, hvordan verden ser ud, afgøre hvilke forskellige handlinger, der er mulighed for at foretage, samt en analyse af hvad konsekvensen af disse handlinger er. På baggrund af denne analyse vælges en handling. Der opbygges et søgetræ af handlinger og den mest optimale handling udføres.

KI'en for et monster i *Doom* skal fungere under helt anderledes forhold. Omgivelserne er dynamiske og spillet afvikles real-time, hvilket betyder at valg for handlinger skal træffes inden for få millisekunder da spillets rytme ellers vil blive brudt, for spilleren vil se monstre "fryse" hver gang et valg skal foretages (og det er hele tiden). Desuden skal der tages højde, for at den enkelt modstander ikke nødvendigvis har information om alle omgivelser og derfor ikke har mulighed for at opbygge et handlingssøgetræ på samme måde som KI'en i en skakcomputer gør.

I afsnit 3.2 nåede frem til, at et udfordrende robotspil med kan tilhøre arkadegenren. Derfor bør de autonome modstanderrobotter kunne operere under vilkår, der ligner dem som monstrene i *Doom* opererer under, så robotterne i et robotspil skal implementeres som autonome agenter. Nareyek beskriver hvordan forskellige agenttyper kam implementeres som autonome modstandere i computerspil [49]. Han nævner en række forskellige agenttyper, hvor nogle vil blive gennemgået i afsnit 4. Det er værd at have i baghovedet, at den agenttype der vælges til spillet, skal vælges med henblik på at skabe en grad af emergens i det endelige robotspil.

Det kan gøre spillet mere uforudsigeligt og dermed også mere udfordrende. En høj grad af emergens er dog ikke nok til at skabe et vellykket spil, for er KI'en for simpel vil man opleve det samme som i Pacman; hvis spilleren bevæger osten på akkurat samme måde i en række spil, så vi spøgelserne i alle tilfælde bevæge sig på samme måde. Derfor skal modstanderene have et abitrært, eller måske endda adaptivt element. Dette vil ligeledes blive behandlet senere.

3.4 Delkonklusion - hvordan kan et godt robotspil se ud?

Målet med denne del af har været at finde ud af, hvilke elementer computerspil består af. Disse elementer kan sammen med de indledende erfaringer med LEGO Spybots give en ide om, hvad der skal til for at skabe sjove og udfordrende robotspil. Opsamles hovedpointerne i det ovenstående vil det være nærliggende at skabe et robotspil i rummet som et actionpræget arkadespil. Derudover skal der være en form for rammefortælling, der kort ridser op, hvorfor den opgave spilleren skal løse overhovedet er interessant at løse. Efterhånden som spilleren opfylder delmål og mål skal denne belønnes på den ene eller anden måde. Har en spiller eksempelvist nedkæmpet en modstander, kan spilleren belønnes med forøget våbenstyrke; med mindre det vurderes, at en nedkæmpet modstander er belønning nok i sig selv. En form for pointsystem kunne også være fornuftigt, idet spilleren så har mulighed for at sammenligne sin præstation med andre spillere - og prale hvis han er bedre.

I den mere tekniske del af spilkonstuktionen er det vigtigt at sørge for, at modspillerne opfører sig uforudsigeligt men fornuftigt, da det vil gøre spillet langt mere udfordrende. Det opnås ved at stræbe efter en høj grad af emergens i spillet og nøje overveje den strategi, der ligger bag modstanderens kunstige intelligens. Hvordan det kan gøres i praksis vil blive behandlet i næste del.

Del II
Analyse af agenter

I foregående del nåede vi frem til at de autonome robotmodstandere og dele af omgivelserne i fysiske robotspil optræder som agenter. Derfor vil den tekniske analyse primært være en undersøgelse af agent generelt og robotter i særdeleshed.

I denne dels første kapitel om agenter, vil en række agenttyper blive gennemgået med henblik på at kunne vælge en agenttype, der vil passe ind i et robotspil.

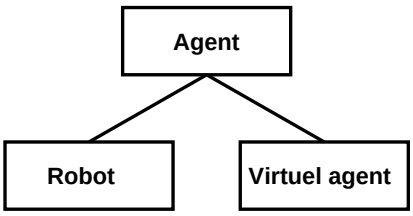
I det efterfølgende afsnit vil en række kommercielle robotprodukter blive gennemgået og en konkret robottype vil blive udvalgt på baggrund af en række nøglekriterier.

4 Agenter

Når de fleste hører begrebet agent kommer de uværligt til at associere det med James Bond og andre agenter fra film, kriminalromaner eller lignende. Og billedet er ikke helt dårligt. En agent er nemlig en enhed, der helt eller delvist kan klare sig selv, imens de søger at opfylde et givet mål.

Agenter kan være af forskellig beskaffenhed; nogle er fysiske og betegnes ofte som robotter; andre er rent softwarebaserede og kaldes virtuelle agenter, software robotter eller kort og godt bots. Vi har allerede set på både fysiske og virtuelle agenter. Spybotten kan optræde som agent, hvis den er blevet udstyret med et passende kontrolprogram. Af software agenter har vi diskuteret de virtuelle agenter, der optræder i computerspil, blandt andet spøgelserne i Pacman og monstrerne i Doom.

For at undgå begrebsforvirring vil jeg i det følgende benytte den distinktion, der er illustreret i figur 4.1, hvor fysiske agenter, der agerer i den fysiske verden benævnes robotter og de rent softwarebaserede agenter, som agerer i den virtuelle verden, kaldes virtuelle agenter.



FIGUR 4.1: Distinktion mellem robot og virtuel agent.

Mange forskere har beskæftiget sig med agenter, ikke mindst de senere, hvor de er blevet gjort til genstand for adskillige undersøgelser. Det har medvirket til, at der er næsten lige så mange definitioner af agent begrebet som der er forskere inden for området.

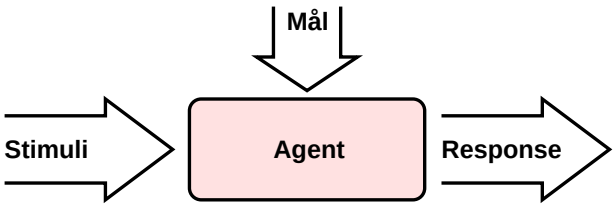
Pattie Maes har en meget bred, men klar definition på hvad en agent er:

> An *agent* is a system that tries to fulfill a set of goals in a complex, dynamic environment. [44] s. 2

En agent er altså en enhed, der forsøger at opfylde en form for mål i komplekse omgivelser. Når der arbejdes inden for et felt som datalogi er en så bred definition sjældent nyttig da både mennesker, dyr, robotter og meget andet falder inden for denne definition.

Jiming Liu har en mere specifik definition på en, hvad en agent er. Han kræver at en agent er en computerkontrolleret enhed og definerer en agent således:

> The notion of agent should be taken in a broad sense, encompassing a wide spectrum of computational entities that can sense their local task conditions and accordingly make decisions on how to react to the sensed conditions by performing certain behaviors in the task environments. [42] s. 2



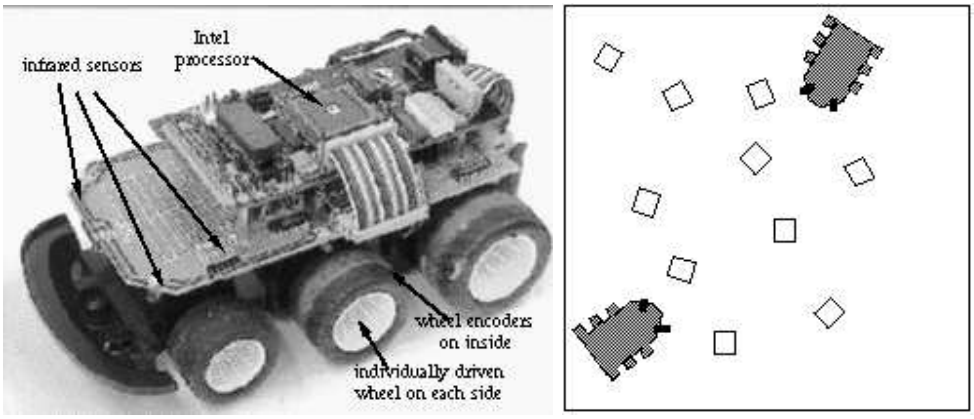
FIGUR 4.2: Skematisk fremstilling af en agent.

En agent som Maes og Liu definerer den, kan illustreres som i figur 4.2. På figuren ses det, at agenten modtager **stimuli** fra omgivelserne via sine sensorer. Stimuli kan for en robot være et signal fra en tryksensor, billeder fra et kamera, lyd, lys eller lignende. Agenter behandler stimuli med henblik på at få opfyldt det eller de mål, som den skal opnå. Målet kan være at finde bolde som i [45], nedkæmpe spillerens aktant som i Pacman eller noget helt andet. Behandlingen af stimuli med henblik på at opnå agentens mål bliver omsat til **respons**, det vil sige handling, i agentens aktuatorer. Har vi fx at gøre med en alarmagent, hvis mål er at overvåge en butik uden for åbningstid, kan agenten være udstyret med en bevægelsessensor og hvis en bevægelse opfanges, kan agenten handle ved at blinke med sine lygter, sige en høj lyd og sende en email til butikkens ejer.

Andre definitioner stiller yderligere krav til agenter, der indgår i multiagent systemer, altså systemer hvor flere agenter indgår. Michael Woolridge foreslår i bogen *An Introduction to MultiAgent Systems*, at agenter ud over at være autonome computerbaserede enheder, ligeledes skal have mulighed for at kommunikere og samarbejde med andre agenter samt koordinere deres arbejde med andre agenter for at kunne opnå deres mål. På den baggrund fremkommer han med følgende definition:

> Agents are computer systems with two important capabilities. First, they are at least to some extent capable of autonomous action - of deciding for themselves what they need to do in order to satisfy their design objectives. Second, they are capable to interact with other agents, not simply by exchaging data, but by engaging in analogues of the kind of social activity that we all engage in every day af our lives: coorporation, coordination and negotiation and the like. [65] preface xi

Woolridge udelader i sin definition de robotter, der samarbejder uden at have kendskab til hinanden og uden at have mulighed for eksplicit kommunikation. Et eksempel på sådanne agenter er Rolf Pfeifers **didabots** [50]. Forrest på robotterne er der to lyssensorer, som peger hver sin retning og der er en vinkel på omkring 90 grader mellem dem. De to sensorer befinder sig omkring ti centimeter fra hinanden. Figur 4.3 viser en didabot samt Pfeifers forsøgsopstilling. Kontrolprogrammet i disse didabots går kort og godt ud på at undgå at støde ind i ting. Når en flok didabots sættes til at køre rundt i et rum uden andre forhindringer end kasser med en størrelse på omkring ti kubikcentimeter, er resultatet at kasserene placeres i to klynger og evt. nogle kasser langs væggene, resten af gulvpladsen er blevet ryddet. Robotterne har altså samarbejdet om at rydde op uden at have noget kendskab til hinanden eller deres opgave for den sags skyld. Didabot forsøget er et godt eksempel på et emergent system, fordi hver enkelt didabot kun forsøger at undgå at støde ind i objekter, men til sammen opnår de noget større, nemlig at rydde op.



FIGUR 4.3: Tv. didabot. Th. Pfeifers opstilling med didabots og kasser. Illustrationen er lånt fra [50]

Figur 4.4 er Woolridges definition af en agent er illustreret. Illustrationen er meget lig figur 4.2 bortset fra, at en del af stimuli kan bestå af kommunikation. Ligeledes kan en del af agentens handlinger være at udsende beskeder. Kommunikation adskiller sig fra andre typer stimuli, da modtageren med sikkerhed ved, at afsenderen er en anden agent. Kommunikation er bearbejdet form for stimuli, for beskeder skal som oftest ikke fortolkes, som det eksempelvis er tilfældet for stimuli fra en tryksensor.

En Spybot er et godt eksempel på en kommunikerende agent. Den ser andre Spybots i sine omgivelser ved hjælp af infrarøde beskeder, som disse udsender og den udsender selv infrarøde beskeder, så andre kan se den. Spybots er i stand til både at sende beskeder til andre Spybots og andre robotenheder fra LEGO.

Stort set alle der skriver om agenter, er dog enige om en ting; nemlig at de

FIGUR 4.4: Skematisk fremstilling af en kommunikerende agent. Illustrationen er lånt fra [49]

skal være autonome. Det vil sige, at de opererer uafhængigt af kontrol udefra fx skal agenten selv kunne afgøre hvordan sensorstimuli skal omsættes til handling i aktuatorene.

4.1 Agent design

Der er en række punkter man bør have for øje når man designer en agent. Neden for er der en oversigt over nogen af de vigtigste:

1. Hvad er agentens målsætning?

2. Hvilket miljø skal agenten agere i?

3. Hvilken beskaffenhed er agenten, det vil sige er den fysisk eller virtuel og hvilke redskaber har den til rådighed?

4. Hvilken type kontrolprogram er passende for, at agenten kan opfylde sit mål?

Første punkt omkring agentens målsætning giver ofte sig selv, da årsagen til overhovedet at beskæftige sig med agenter ofte er, at man har en opgave, der skal løses og man mener at en agent af en eller anden type kan klare opgaven.

Derpå er det vigtigt at gøre sig klart, hvilket miljø agenten skal operere i. Skal den kunne støvsuge, være modstander i et computerspil, foretage booking af hotelværelser eller undersøge overfladen på Mars. En robot, der skal undersøge Mars skal være langt mere robust end en støvsugerrobot - til gengæld må den også koste en milliard mere. Tilsvarende er stabilitet og pålidelighed for en virtuel agent til booking af hotelværelser vigtigere end for en agent i et computerspil.

Sidst punkt man skal overveje når man designer en agent er, hvilken type kontrolprogram der er passende til denne type agent. Forskellige typer af kontrolprogrammer til agenter gennemgås i næste afsnit.

Benyttes James Bond igen som eksempel er han en fysisk agent, målsætningen er ofte at redde verden ved at eliminere superskurken og han skal kunne agere i hele verden. Han har alle *Q*'s opfindelser til rådighed.

Helt så komplicerede opgaver skal computerbaserede agenter sjældent løse. Ikke desto mindre er alle punkterne vigtige at holde sig for øje, hvis det skal lykkes for agenten at løse sin opgave. Er der tale om en robot, der skal slå græs, skal den ud over at være fysisk, være udstyret med en form for klippeanordning samt være i stand til at afgøre, hvilket græs der skal slås. Desuden bør dens kontrolprogram kunne tage højde for, at græsplænens ejere har efterladt en havestol og en fodbold i forbindelse med løsningen af dens opgave. Figur 4.5 viser en plæneklipper- og en støvsugerrobot.

FIGUR 4.5: Tv. plæneklipperrobot fra frindly robots [7]. Th. robotstøvsugeren Trilobite fra Electrolux [15].

4.1.1 Ydre adfærd

I afsnit 3.3.2 blev kunstig intelligens i forbindelse med spil behandlet. Det blev nævnt at programmører og spillere af computerspil er ligeglade med, om agenter i computerspil er intelligente, så længe deres adfærd er fornuftig. Derfor er det, i forbindelse med en analyse af robotter til spil, nyttigt at behandle agenters **ydre adfærd** og deres kontrolprogram, hver for sig. Den ydre adfærd beskæftiger sig udelukkende med, hvordan agenten rent empirisk agerer i sine omgivelser hvor det er kontrolprogrammet, der ligger til grund for den ydre adfærd.

Vi ønsker en spilrobot, hvis ydre adfærd opfylder de kriterier, der er blevet opstillet nedenfor. Med disse kriterier for øje er det lettere at vælge et kontrolprogram, der muliggøre at kravene til spilrobotternes ydre adfærd opfyldes.

1. Agenterne skal opføre sig fornuftigt.

2. Agenternes opførsel bør være uforudsigelig.

3. Agenterne bør være i stand til at samarbejde.

Agenterne skal opføre sig fornuftigt, da ufornuftige og irrationelle handlinger skaber forvirring hos spilleren. Hvis modstanderne i et arkadeskydespil eksempelvis løber rundt og skyder på hinanden, eller ikke kan finde aktanten selv om denne står lige foran dem, så er det med til at ødelægge spilleglæden en smule. Det påpeges også af spildesignernen Richard Rouse [53]. Uforudsigelighed er hensigtsmæssigt, da spillet da vil være mere udfordrende end hvis man kan forudsige sin modstanders næste træk [31]. At agenterne bør være i stand til at samarbejde, er igen en konsekvens af den kritik, brugerene gav spillene, der indgik i de første spilscenarier.

4.2 Agent typer

En agents ydre adfærd er betegnelsen for det, man oplever, når man ser på den. Man oplever ofte at folk, der betragter robotter, beskriver deres adfærd med de samme ord som bruges til at beskrive dyr og menneskers handlinger, på trods af at den, der observerer udemærket godt ved, at de processer, der ligger til grund for handlingerne langt fra er de samme, som der ligger til grund for dyr og menneskers handlinger [19].

Kontrolprogrammet er betegnelse for det system, der afføder de ydre handlinger. Der er ikke nødvendigvis en direkte sammenhæng mellem ydre handlinger og kontrolprogrammet, forstået på den måde, at et simpelt kontrolprogram godt kan give anledning til forholdsvist avanceret adfærd, mens komplicerede og avancerede kontrolprogrammer godt kan føre til simpel adfærd. Eksempelvis udviser didabots under de rigtige omstændigheder en fornuftig adfærd og omvendt kan en supercomputer på metrologisk institut være i gang med en stor mængde langhårede udregninger uden den ydre adfærd ændre sig.

Når vi ytrer ønske om at skabe agenter til spil, der opfører sig fornuftigt så stiller vi udelukkende krav til den ydre adfærd. Den tilgang adskiller sig væsentligt fra den som mange forskere i kunstig intelligens havde og nogle stadig har. Inden for det område er fokus på kontrolprogrammet; fx hvordan får vi vores agent til at foretage fornuftige valg. Når vi har med agenter i spil at gøre, er det vigtigste om agentens ydre adfærd kan udfordre spilleren i passende grad.

Fornuftig adfærd i spil kan opnås på flere måder og i det følgende vil tre forskellige strategier til kontrolprogrammer blive præsenteret. Jeg vil redegøre for, hvilken ydre adfærd man vil kunne opnå ved den pågældende strategi og sidst men ikke mindst give eksempler på, hvordan de er blevet benyttet i spil. De tre agent strategier er; **reaktive agenter** [65], **tilstands agenter** [49] og **adaptive agenter** [44].

4.2.1 Reaktive agenter

Begrebet *reaktiv* i forbindelse med agenter bliver benyttet på flere forskellige måder. Jeg vil benytte reaktiv på samme måde som Nareyek, der benytter det om agenter, som har en direkte forbindelse fra stimuli til respons [49]. Det betyder når vi snakker om ydre adfærd, at den samme agent altid reagerer på samme måde på samme stimuli. At agenten altid reagerer på samme måde på stimuli af samme type betyder, at agenten betragtet udefra kun har en tilstand[1]. Eksempler på robotter af denne type er Pfeifers didabots [50], eller Braitenbergs væsner [30].

For computerspil er der især blandt de ældre spil, flere agenter af denne type. Som eksempler kan nævnes den tilfældighedsgenerator, der i *Tetris* afgører, hvilken figur der skal komme næste gang. Den tager ingen hensyn til, hvordan det går for spilleren, hvilken figur der kom sidst eller noget som helst andet.

Af arkadespil kan nævnes det klassiske *Space Invaders*, hvor de invaderende horder fra rummet bevæger sig ned imod jorden uden hensyntagen til, hvor spillerens aktør befinder sig befinder sig.

Reaktive agenter som beskrevet ovenfor er som udgangspunkt for simple til at leve op til de krav, vi har stillet til robotter i arkadeinspirerede spil i det fysiske rum. Reaktive agenter kan godt have en fornuftig opførsel selv i komplekse omgivelser, men det er sværere at give dem en fornuftig adfærd i forhold til den aktant, der styres af en menneskelig spiller.

I figur 4.6 er der en skitsering af, hvordan en simpel reaktiv agent i et skydespil kunne være implementeret. Det eneste robotten gør er at køre tilfældigt rundt og skyde til højre og venstre, og skelner ikke mellem ven og fjende - eller noget som helst andet for den sags skyld.

```
start  program evilReactiveAgent
   forever do
     moveRandom
     shoot
   end forever
end program
```

FIGUR 4.6: Skitsering af kontrolprogram til en reaktiv agent i et skydespil

.

[1]Det skal nævnes at tilfældighedsgeneratorer kan benyttes i en reaktiv agent. Derfor betyder det at reagere på samme måde ikke det samme som at udføre samme handling, men at der reageres efter samme mønster uden hensyntagen til robottens omgivelser og hvad den med fordel har gjort i lignende situationer

4.2.2 Tilstandsagenter

Agenter med tilstand, som Woolridge betegner dem [65], eller dem som Nareyek kalder *triggering agents* i [49]; betegner begge agenter, der minder om de reaktive, men hvor flere tilstande er indført. Disse agenter agerer forskelligt alt efter, hvilken tilstand agenten befinder sig i og de skifter strategi når bestemte hændelser forekommer. Støvsugerrobotten Trilobite i figur 4.5 opfører sig på nogle områder en tilstandsagent. Dens opgave er at støvsuge, men er Trilobite ved at løbe tør for batteri under en støvsugning, skifter den tilstand og kører hen til sin batterioplader for at lade op, hvorefter støvsugningen genoptages.

Et klassisk eksempel på en tilstandsagent er ELIZA; en virtuel psykiater, der blev konstrueret med det formål at overbevise omverdenen om, at det var muligt at konstruere en simpel virtuel, agent der alligevel forekommer at være fornuftig ved første øjekast. I kølvandet på ELIZA opstod der en række lignende computerprogrammer. Nedenfor følger et citat, der er et uddrag af en dialog mellem Parry en simuleret paranoid og Doktor en simuleret psykiater. Samtalen fandt sted 18 september 1972. Parrys udtalelser starter med *.

> Good Evening.
>
> Tell me your problems. Please terminate input with a period or a question mark.
>
> *People get on my nerves sometimes.
>
> I am not sure I understand you fully.
>
> *You should pay more attention.
>
> Suppose you should pay more attention.
>
> *You're entitled to your own opinion.
>
> What makes you think I am entitled to my own opinion?
>
> *It bothers me just to be around people in general.
>
> Please go on.
>
> *I went to the track at Bay Meadows a while back.
>
> What does that suggest to you?
>
> *I went to the races.
>
> What else would you like to discuss?
>
> *Horseracing interests me.
>
> Why do you say that just now?
>
> *There's a lot of crooked gambling going on there.
>
> I am not sure I understand you fully,

[...]

*Goodbye.

It's been my pleasure, that's $399.29 please.

[22]

Samtalen virker ved første øjekast fornuftig nok til at kunne stamme fra en virkelig konsultation. Det algoritmen bag Doktor blandt andet gør, er at lede efter nøgleord, som *you, father, mother* i sin samtalepartners sætninger, udskifte nøgleordene med ord, der ville være passende i en svar fx *I, me* eller *family* og herefter benytte resten af sætningen i sit næste spørgsmål. På et tidspunkt siger Parry *You're entitled to your own opinion*, hvortil Doktor svarer *What makes you think I am entitled to my own opinion?* Men kan Doktor ikke finde nøgleord benyttes standardsætninger som *I am not sure I understand you fully*, *Please go on* og *Why do you say that just now?*

Eksemplet med samtalen mellem Doktor og Parry illustrerer mulighederne og begrænsningerne ved tilstandsagenter. Deres adfærd kan være varieret og til tider overraskende, men i løbet af kort tid vil man alligevel gennemskue et mønster i adfærden, hvorefter den vil forekomme for triviel og kedelig.

Samtalen er et fint eksempel på et emergent system, for her udviser to simple agenters adfærd tilsammen noget der til forveksling kan ligne menneskelig intelligens.

I computerspil er spøgelserne i *Pacman* eksempler på simple tilstandsagenter. De har nemlig to tilstande; aggressive når de jagter osten gennem labyrinten, og bange når Pacman har spist en energipille, der gør ham i stand til at æde spøgelserne.

Inden for computerskydespil kan der ligeledes findes eksempler især i de tidligere spil. Nazisterne i *Wolfenstein 3D* er også agenter med tilstande. De går rundt og patruljerer indtil de får øje på spillets hovedperson B. K. Blazkowicz, hvorefter de går til angreb.

Autonome Spybotmodstandere programmeret i LEGOs grafiske programmeringsmiljø hører til denne kategori. Det Spybotprogram der blev benyttet i spillet, som blev beskrevet i afsnit 5.6 kan groft skitseres som i figur 4.7.

4.2.3 Adaptive agenter

Det kræver en dygtig programmør at programmere en agent med en fornuftig adfærd, ud fra en reaktiv- eller tilstandsagent strategi. Vi ved fra Pattie Maes definition af agenter, at de skal kunne gebærde sig i komplekse omgivelser.

Alle der har programmeret robotter har oplevet at uforudsigelige situationer opstår. Programmering af agenter vil være langt lettere, hvis det var muligt at programmere nogle læernemme agenter, der kunne tilpasse deres mål til verden omkring dem. Og det er netop, hvad de adaptive agenter gør. De tilpasser sig de

```
start program evilSpybot
   forever do
      if otherSpybotIsClose then
        shoot
      else if otherSpybotIsSeen then
        moveTowardsOtherSpybot
      else if otherSpybotNotSeen then
        searchArea
      end if
   end forever
end program
```

FIGUR 4.7: Skitsering af kontrolprogram til modstander Spybot med flere tilstande

omgivelser, som de opererer i. Pattie Maes betegner en agent som adaptiv, hvis den er i stand til at forbedre sig over tid, hvilket vil sige, at den bliver bedre til at opnå sine mål efterhånden som den får tilpasset sig det miljø, den befinder sig i [44]. Der findes flere strategier, der kan implementeres for at opnå adaptivitet, blandt andet en evolutionær strategi og neurale netværk. I det følgende vil jeg gennemgå hovedtrækkene i en af disse strategier. Senere vil det blive overvejet, hvilken tilgang er mest fornuftig i forbindelse med implementering af adaptive robotter i robotspil.

Evolutionær strategi Evolutionære algoritmer eller evolutionære strategier er, som navnet antyder, biologisk inspireret. I forbindelse med agenter kan teknikken benyttes til udvikling af et kontrolprogram, og/eller til at justere adfærde og variabler i et kontrolprogram. Kernen i de evolutionære algoritmer er gener, fitnessværdi, mutation, flere individer, flere generationer og eventuelt krydsning mellem individer [38, 48].

Generne skal indeholde de egenskaber, man ønsker at forbedre. Det kan fx være værdier til løsning af en ligning. I forbindelse med agenter kan evolutionære algoritmer benyttes både til at udvikle kontrolprogram og til at tilpasse (tune) variabler til bestemt adfærd i et bestemt miljø [33, 47, 23]. Jakob Fredslund benyttede i sit speciale begge aspekter for at få en robot til at følge en streg [24].

Fitnessværdien bliver beregnet på baggrund af en fitnessfunktion, der udtrykker hvor godt dette individ klarer sig. Ønsker man at finde en god løsning til en ligning er fitnessværdien et meget præcist udtryk for, hvordan et individ klarer sig. Men for en fysisk robot som en stregfølgerobot, hvor fitnessfunktion tager udgangspunkt i fysiske målinger, så som information om hvor længe robotten har fulgt en streg og hvor længe den har kørt uden-

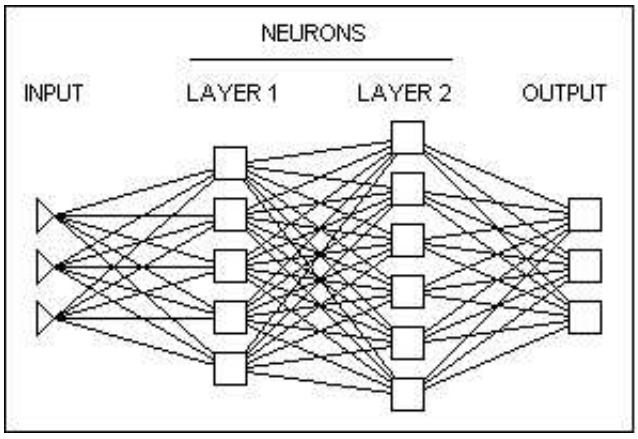
for, samt om den har kunnet finde tilbage til stregen hvis den er kørt af. At følge stregen og finde tilbage til den, vil i dette eksempel give en højere fitness værdi, hvorimod det vil trække fra i fitnessværdien hvis robotten kører af stregen. Men der vil være større usikkerhed på fitnessværdien for en fysisk robot, fordi robotten vil kunne måle forkert.

For hver generation evalueres fitnessværdien for hvert individ i en population. Herefter gemmes de bedste individer og de resterende individer tildeles nye gener, der fremkommer på baggrund af krydsning af generne fra de bedste individer[2]. Individerne i den nye population får muteret deres gener med en vis sandsynlighed.

Når en population har udviklet sig over en række generationer, bør der være mindst et individ, der har løst den opgave der som udgangspunkt havde skulle løses.

Det adaptive element opstår fordi der for hver generation frasorteres individer med en lav fitnessværdi, og det er de individer der løser opgaven dårligst. Efter nogle generationer vil der (forhåbentligt) være et eller flere superindivider, som takket være de tidlige generationers erfaringer løser opgaven tilfredsstillende.

Adaptivitet baseret på neurale netværk Neurale netværk, som det illustreret i figur 4.8, er grafer med et lag bestående af en eller flere input-knuder, til venstre på figuren, og et lag bestående af en eller flere output-knuder, til højre på figuren. I mellem input og output er der nul eller flere lag af knuder der kaldes **skjulte lag** eller **hidden layers** som er den engelske betegnelse [48].



FIGUR 4.8: Neuralt netværk.

[2]Det skal dog bemærkes at ikke alle evolutionære algoritmer benytter sig af krydsning mellem individer.

Hver knude har en tærskelværdi og hver kant har en vægt. Får en knude input fra kanter med en samlet sum større end dens tærskelværdi "fyrer" den, hvilket vil sige at den sender et output til det næste lag.

Et neuralt netværk lærer ved, at det gives nogle input, hvor man på forhånd ved, hvilket output man ønsker. Denne type input kaldes **tranining-data**. På baggrund af hvert sæt traning data justerer man netværkets vægte så outputtet bliver som man ønsker. Denne træning kan foretages på en række forskellige måder blandt andet en teknik, der kaldes **back-probagation**. Efter træningen gives netværket endnu en række traning-data for at kontrollere, om netværket giver det ønskede output på ukendt input; netværkets vægte trænes ikke i denne fase.

Inden for adaptive robotter vil man typisk give et input for hvert af robottens sensorer og et output for hver aktuator (motor, højtaler eller lignende). Gennem træning skal man sørge for at robotten kommer til at handle korrekt på baggrund af en given handling.

Problemet ved brug af neurale netværk er blandt andet at skaffe en passende mængde træningsdata. Neurale netværk er desuden mest velegnede til at løse en bestemt nøje defineret opgave, som for eksempel at analysere et billeder fra mamografier for at identificere brystkræft [40], eller finde lys som robotterne i [63, 23], frem for at skulle løse bredere opgaver som *find en modstander og nedkæmp den, bliver du såret så træk dig tilbage.*

4.3 Samarbejdende agenter

Laver man et kontrolprogram til en modstanderrobot ved hjælp af det grafiske værktøj som Spybotics stiller til rådighed så får man Spybot modstandere, som opfører sig uhensigtsmæssigt. Kan ikke se forskel på "ven" og "fjende" og de er ikke i stand til at samarbejde, hvilket ellers kunne gøre spillet langt mere udfordrende.

Et samarbejde kunne bestå af at lade en modstanderrobot, der fik øje på spiller udsende en alarmbesked til de andre modstander om, hvor spilleren var blevet set. Derpå kunne modstanderne koordinere deres angreb på spilleren. Skal autonome robotmodstanderes handlinger koordineres, er det umiddelbart nærliggende, at skele til hvordan koordinering af virtuelle modstandere foregår i computerspil.

I den virtuelle verden, hvor hver agent har mulighed for global information om alle andre agenters placering, er det eksempelvist forholdsvist enkelt at skabe en flokadfærd [52], hvilket kunne være nyttigt, hvis flere modstander skulle angribe samtidigt. Det kan ligeledes lade sig gøre at få agenter til at bevæge sig problemløst rundt i en virtuel verden [51]. Men i den fysiske verden er der sjældent mulighed for pålidelig global information.

Samarbejde mellem robotter forudsætter som oftest kommunikation mellem disse. Hvordan kommunikationen og samarbejdet mellem robotterne skal foregå

kommer helt an på den opgave, de skal løse. Derfor er det nødvendigt at finde en passende opgave for modstanderrobotterne.

Da de skal indgå i et spil, kan det antages, at omgivelsernes udformning er tilpasset robotternes egenskaber; en antagelse man ikke nødvendigvis kan tillade sig, hvis robotterne skal løse opgaver i den virkelige verden. Samarbejde mellem robotterne kan være implicit, hvor robotterne ikke kommunikerer, eller eksplicit hvor de kommunikerer for at løse deres opgave.

Eksempler på grupper af robotter der samarbejder implicit for at løse deres opgave uden at kommunikere, er Pfeifers didabots [50], eller Fredslunds flokke af robotter, der kører i formation ved hjælp af lokal information [25].

Et eksempel på en gruppe af robotter, der løser deres fælles opgave med brug af eksplicit samarbejde er Berkey og Mataric's tre robotter, der arbejder sammen om at om at flytte en kasse [27]. En robot koordinerer arbejdet og to andre byder ved hjælp af et auktionssystem ind på at løse opgaven, hvilket medfører en del kommunikation mellem robotterne.

Implicit og eksplicit samarbejde har hver deres fordele og ulemper. For begge gælder, at antallet af tilstande i systemet stiger eksponentielt med antallet af agenter. Har en agent T tilstande og indgår der a agenter i systemet, er det samlede antal tilstande i systemet T^a, hvilket hurtigt bliver uoverskueligt [46]. I grupper af robotter, som benytter sig af eksplicit samarbejde stiger mængden af kommunikation, når antallet af robotter øges [46], hvilket er et problem, da der altid er begrænset båndbredde til kommunikation. Grupper, der samarbejder implicit har ikke problemer med båndbredden men til gengæld er de globale konsekvenser af de enkelte robotters lokale adfærd svær at forudsige [46]. Der er med andre ord tale om et dilemma, hvor brugen af implicit eller eksplicit kommunikation skal afgøres på baggrund af den opgave robotterne skal løse.

Skal robotter løse en opgave, er det som udgangspunkt lettest gjort ved at benytte kommunikation mellem robotterne. Til at kunne formidle informationer mellem de eksplicit samarbejdende robotter, kan **ad hoc netværk** være et nyttigt redskab.

4.3.1 Ad hoc netværk

Ad hoc netværk er netværk hvor knuderne er mobile [41, 64].

> The networking literature defines ad hoc, or multi-hop mobile wireless network, as a network with a set of geographically dispersed nodes in which each node is able to forward packets for other nodes that cannot communicate directly. Thus each node is able to act as a router, as well as a data source, or sink [64] s. 2.

Alan Windfield, beskæftiger sig med indsamling af data ved hjælp af mobile trådløse ad hoc netværk. Han har konstrueret en simulator, hvor en flok kortlægningsrobotter bevæger sig rundt i et afgrænset område, undersøger stedet de

befinder sig på og sender datapakker med resultatet fra undersøgelsen ud i netværket. Netværket består af de andre robotter, der skal sørge for at pakken bliver sendt videre indtil den når et fast opsamlingspunkt, hvor data for hele området opsamles [64].

I robotspil vil Windfields fremgangsmåde kunne benyttes af modstanderrobotter, så de kunne udveksle information om spilleren eller spillernes position.

Et karakteristika ved mobile ad hoc netværk er, at det er stort set umuligt at forudsige netværkstopologien på et givet tidspunkt, så derfor kan de mere traditionelle netværkskommunikationsprotokoller ikke benyttes.

Robotterne i Windfields artikel er baseret på radiokommunikation der giver en lige stor sendeafstand til alle sider. På baggrund af dette samt antagelser omkring størrelsen af det område, der skal undersøges, simulerer han hvordan netværket udvikler sig over tid. Netværket bliver mindre og mindre sammenhængende over tid, men det betyder ingenlunde, at datapakkerne ikke ender ved opsamlingspunktet. For da knuderne i netværket hele tiden bevæger sig vil datapakkerne med tiden bevæge sig tilbage imod opsamlingspunktet. Undersøgelserne viser ligeledes at jo flere robotter, der befinder sig i det afgrænsede område der skal undersøges, jo større sammenhængende delnetværk er der til stede.

Windfield gør ikke rede for, hvorledes robotterne, der kun besidder lokal information skal kunne holde sig inden for et afgrænset område. Han antager at robotterne har hukommelse nok til at huske tilstrækkelig mange beskeder og desuden at robotterne er udstyret med synkroniserede ure, hvilket er en farlig antagelse da alle distribuerede systemer i praksis er asynkrone [26].

Alt i alt er Windfields undersøgelser et godt eksempel på et system, der er velfungerende som simulation men sandsynligvis ikke vil kunne omsættes direkte til praksis. Om det overhovedet giver særlig god mening at skabe simulatorer for at afgøre, hvorvidt en opgave vil kunne løses i praksis er et emne der er blevet diskuteret voldsomt blandt andet inden for udviklingen af adaptive robotter. Næste afsnit vil behandle nogle af forskellene på robotter og virtuelle agenters vilkår.

4.4 Robotter og virtuelle agenters vilkår

Man kan lige så godt slå det fast først som sidst: det er noget bøvl at arbejde med robotter. Bare det at anskaffe den rigtige type robot og måske endda i større mængder kan være problematisk. Virtuelle agenter er lettere at arbejde med, de bruger ikke batteri, de går ikke i stykker og har man konstrueret én, kan de kopieres i hundredevis, uden at det koster ekstra.

Derfor har flere eksperimenteret med at udvikle robotkontrolprogrammer på virtuelle agenter i virtuelle omgivelser og herefter overføre de udviklede kontrolprogrammer til de fysiske agenter.

I forhold til at udvikle den perfekte robotmodstander til et spil er det umiddelbart fristende at udvikle et kontrolprogram på en PC, overføre det til en fysisk

robot og spille spillet. Men kan det lade sig gøre og kan det i givet fald betale sig?

Nick Jakobi et al. gør i artiklen *Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics* [32] rede for, hvorledes det er muligt at skabe en realistisk simulatior ved at tilføje en tilpas mængde støj. Den tilføjede støjen skal have baggrund i målinger af støj i de fysiske omgivelser. Påstanden er, at en agent der agerer i simulatoren minder meget om tilsvarende agenter i den fysiske verden. Jakobi gør opmærksom på, at det er en omstændelig proces at skabe en sådan simulator, men at processen giver den fordel i forbindelse med evolutionært udviklede robotter, at hundredevis af generationer kan afvikles på ganske kort tid, hvilket ikke ville være tilfældet, hvis fysiske robotter skulle gennemleve hundredevis af generationer. Jakobi et al. viser, at deres tese om, at robotkontrolprogrammer udviklet i simulatorer med en tilpas mængde støj, kan overføres direkte til robotter, er plausibel. Det gør de ved at overføre et kontrolprogram udviklet ved hjælp af en simuleret Khepera robot til en fysisk Khepera robot samt konstatere at programmet opfører sig som forventet.

Mataric og Cliff mener grundlæggende, at det skal kunne betale sig at betjene sig af en simulator, hvis en sådan skal benyttes [47]. Det betyder, at det enten skal være hurtigere at udvikle simulatoren og udvikle kontrolprogrammet end at skrive kontrolprogrammet i hånden, eller også skal programmet udviklet i simulatoren have en markant bedre ydelse. De konkluderer, blandt andet rettet mod Jakobi et. al., at ingen af disse to krav er blevet opfyldt af simulatorer hidtil.

Som respons på Mataric og Cliffs artikel skriver Ficici, Whatson og Pollack en artikel, hvor de præsenterer deres bud på, hvordan robotkontrolprogrammer kan udvikles evolutionært ved hjælp af en teknik som de kalder **Embodied Evolution** [23]. Deres forsøg går kort fortalt ud på, at en gruppe robotter skal udvikle deres gener, så de bliver i stand til at finde lys. Som udgangspunkt er alle deres genre nul, hvilket betyder, at de bare står stille. På banen er robotter med tilfældige faste gener og robotter med faste håndkodede gener, der kan løse opgaven. Hver robot starter med en vis mængde energi og de sender muterede versioner af deres gener til omkringstående robotter med en frekvens, der er proportional med deres energiniveau. Energiniveauet aftager hver gang en robot har sendt gener. Hver gang en robot finder lyset får den ekstra energi og dermed vil de robotter, der finder lyset oftest sende deres gener oftere (energi benyttes som en form for fitnessfunktion). Jo højere en robots energiniveau er, jo mindre er den tilbøjelig til at modtage sendte gener [23].

Forfatterne mener selv, at *Embodied Evolution* med fordel kan benyttes blandt andet, hvor agenterne er nødt til at lære i det fysiske miljø og ved løsning af opgaver i komplekse interaktive miljøer, hvor der fx opstår emergent gruppe eller holdadfærd.

4.4.1 Virtuelle agenter

Virtuelle agenter spænder fra chat-bots til autonome modstandere i computerspil. De har dog alle en ting til fælles; de bliver ikke forstyrret af ydre påvirkninger, med mindre der slukkes for computeren de kører på. De omgivelser, agenterne skal agere i, er fri for støj og desuden er de velkendte og gennemskuelige for programmøren, ikke mindst fordi at denne ofte selv har været med til implementere omgivelserne.

Det er en ganske overkommelig opgave at skrive et kontrolprogram til en agent i et spil (afhængig af spillets kompleksitet), for programmøren har fuld kontrol over agentens omgivelser og har agenten problemer med sine omgivelser kan disse tilpasses til agenten. Desuden har agenten adgang til global information om omgivelserne samt om andre agenter.

4.4.2 Robotter

I modsætning til den virtuelle verden, så er den fysiske verden kompleks og uforudsigelig. For det første ligger man som programmør under for en lang række fysiske forhindringer, som man ikke møder i den virtuelle verden som fx tyngdekraften, tilfældig støj og uforudsigeligheder, som motorer der brænder sammen, batterier der løber tør for strøm og robotdele der falder af robotten.

Når et stykke software skal udvikles til og implementeres i et konkret stykke hardware er det ofte nødvendigt at gå på kompromis når koden skrives. Skriver man kode til en Khepera robot nytter det ikke, at der i koden antages, at robotten på baggrund af omdrejningstællerne på hjulene ved præcist, hvor den befinder sig, da omdrejningstællerne er for unøjagtige til dette formål. Det er sjældent nødvendigt at indgå kompromiser, når man arbejder med virtuelle agenter for den virtuelle agent kan udbygges med sensorer eller tilpasses på andre måder efter programmørens ønske.

4.5 Opsummering

Tabel 4.1 viser en oversigt over de største forskelle på vilkårne for robotter og virtuelle agenter.

På baggrund af dette kapitel mener jeg, at den bedste bud på et kontrolprogram til en autonom modstanderrobot, der skal indgå i en prototype til et robotspil, er at håndkode et kontrolprogram men give mulighed for, at agenten kan tilpasse sig sin modspillers strategi ved at implementere en evolutionær algoritme. Man kunne lade agenten tilpasse sig på en måde der minder om *Embodied Evolution*. Den tid, der kunne være benyttet på at udvikle en simulator til de indledende afprøvninger, mener jeg er bedre brugt på at udvikle selve kontrolprogrammet.

Det vil sandsynligvist være lettest få agenter til at samarbejde eksplicit, så

Robotter	**Virtuelle agenter**
Ringe kontrol over omgivelserne	Fuld kontrol over omgivelserne
Lokal information	Global information
Meget støj	Ingen støj
Kompleks verden	Simpel verden
Agent hardware problemer	Ingen agent hardware problemer
Fysiske begrænsninger	Ingen fysiske begrænsninger
Begrænset processorkraft, hukommelse etc.	Mere processorkraft, hukommelse etc.

TABEL 4.1: Oversigt over de vigtigste forskelle mellem fysiske og virtuelle agenters vilkår i spil

derfor skal agenterne være udstyret med kommunikationsudstyr. At benytte kommunikerende agenter har dog den ulempe, at antallet af modstandere ikke vil kunne skaleres op ubegrænset, da mængden af kommunikationen vil stige, når antallet af robotter stiger. Dermed er det også muligt at ideen om at implementere ad hoc netværk i modstanderrobotteren må droppes.

5 Robotter og Spybots

I det sidste kapitel blev en række af aspekter ved agenter gennemgået. Robotter blev behandlet som en del af kapitlet. På baggrund af hvad der hidtil er blevet gennemgået om spil og agenter, er det nu på tide at afgøre, hvilken type robotspil det kunne være interessant og samtidigt muligt at implementere. Vi nåede frem til, at robotterne i et robotspil burde kunne kommunikere; derfor vil aspekter af de mest udbredte kommunikationsformer til robotter blive gennemgået.

Derpå vil en række kommercielle robotprodukter blive gennemgået. På trods af at spillet bliver baseret på Spybots da disse er tilgængelige i stort antal, vil der blive valgt en eller flere kandidater, der kan indgå i robotspil. Det gøres for at vurdere om Spybots er det bedste bud på robotter, der kan indgå som aktøre i et robotspil.

Til sidst vil Spybotproduktet blive gennemgået i større detalje og nogle erfaringer med at konstruere robotspil ved hjælp af Spybots og Spybotics produktet vil blive gennemgået.

5.1 Diskussion af robotspil

Robotspil, jeg synes der kunne være udfordrende at konstruere er spil, hvor spilleren selv har mulighed for at interagere direkte med robotterne. Et robotbaseret Pacmanspil kunne bestå af en kæmpe labyrint og fire spøgelsesrobotter i menneskestørrelse eller større. Spillerens opgave kunne, lige som ostens, være at samle point samt undgå at støde ind i spøgelserne.

Et endnu vildere og mere actionprægede spil kunne være et robotgladiatorspil hvor spilleren kæmpede på liv og død med svært bevæbnede kamprobotter.

Skal man holde sig til spil med mindre robotter, hvor aktanten bliver styret af en spiller, som dermed selv kun deltager i spillet via den aktant han styrer, er der stadig en række muligheder for et spils udformning. Et spil kan omfatte en eller flere spillere og en eller flere robotter. Det vil være en fordel, hvis spillet kan spilles af både af en og flere personer, på samme måde som klassiske computerspil ofte har mulighed for at blive spillet af en eller flere personer.

Typen af robotspil, der kan implementeres, begrænses af den eller de robotter man vælger at basere spillet på. Der er to tilgange til design af robotspil:

1. Man kan udtænke det mest fantastiske robotspil, der kan implementeres, hvis den rigtige type robotter er til stede, som det blev gjort ovenfor.

2. Man kan tage udgangspunkt i de robotprodukter, der er tilgængelige og herefter finde ud, af hvilken type spil de vil passe ind i.

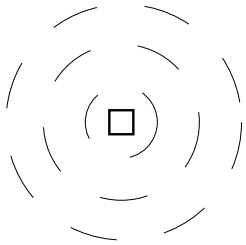
Det er sjovt at udtænke fantastiske og underlige robotspil, men det sjove blegner noget, hvis ideerne ikke kan omsættes til praksis. Derfor vil jeg starte med at vælge et eksisterende robotprodukt, som kan indgå i en robotspilsprototype.

5.2 Robotter og kommunikation

Skal robotter fungere fornuftigt i et robotspil er kommunikation mellem dem næsten uundværlig. Når robotter kommunikerer er de to mest udbredte former for udveksling af beskeder kommunikation baseret på radiobølger og infrarøde (IR) bølger. I dette afsnit vil disse to former for kommunikation mellem robotter blive gennemgået. Fordele og ulemper ved de to forskellige kommunikationsformer vil trukket frem.

5.2.1 Radiokommunikation

Radiokommunikation foregår vha. radiobølger; det vil sige bølger i frekvensområdet op til $10^9 Hz$ [28]. Radiobølger udbreder sig i en kugle fra senderen på samme måde som ringe i vandet, se figur 5.1. Signalet er stærkest ved kilden (afsenderen) og aftager jo længere man bevæger sig sig væk fra kilden. Robotter der benytter denne form for kommunikation vil typisk have mindst en radiobølge sendeenhed og mindst en radiobølge modtagerenhed.



FIGUR 5.1: Udbredelsen af radiobølger.

Radiobølger har den fordel, at de gennemtrænger de fleste materialer, hvilket giver mulighed for at kommunikere på trods af forhindringer, på tværs af rum og også over lange afstande.
Fordele ved radio kommunikation:

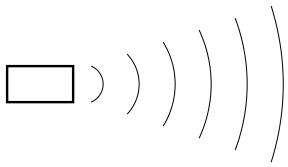
- Det er muligt at kommunikere i mellem flere rum.
- Robust i forhold til ydre påvirkning.
- Mulighed for at kommunikere retningsbestemt.
- Mulighed for afstandsbedømmelse.

Ulemper ved radiokommunikation:

- Dyrere end IR kommunikation.

5.2.2 IR kommunikation

IR kommunikation foregår ved hjælp af infrarøde bølger i frekvensområdet $3 \cdot 10^{11} Hz$ til $4 \cdot 10^{14} Hz$ [28]. Robotter, der benytter denne kommunikationsform har som hovedregel mindst en IR sender og mindst en IR modtager. Da infrarøde bølger er lys i det infrarøde område har de mange egenskaber tilfælles med naturligt lys, der indholder en del IR bølger. Det betyder blandt andet, at IR kommunikation aldrig forlader et lukket rum, det vil sige rum uden vinduer, sprækker og åbne døre. Infrarødt lys reflekterer på visse overflader som glatte lyse, men absorberes af sorte ujævne overflader. Da naturligt lys som nævnt indeholder IR lys, bliver infrarøde beskeder let forstyrret af støj fra omgivelserne, især sollys og lys fra lysstofrør er forstyrrende for IR kommukikation. Det kan også være en ulempe, at IR bølger reflekteres på nogle overflader og absorberes af andre. Fx kan man forhindre en sender og en modtager, der står 10 cm fra hinanden, i at kommunikere, blot ved at stikke et stykke sort karton i mellem dem.



FIGUR 5.2: Udbredelsen af infrarøde bølger.

Det er muligt at sende signaler i en bestemt retning på samme måde som et spotlight gør med naturligt lys. Det giver modtageren mulighed for at bestemme i hvilken retning, afsenderen befinder sig og samtidigt giver det mulighed for at sende beskeder til modtagere i en bestemt retning, uden at andre modtagere kan opsnappe beskeden.

Der er lavet en række eksperimenter omkring IR kommunikation mellem autonome agenter. Blandt andet har både Kaspar Støy og Lionel Sachs et al. undersøget dette område. Begge tager udgangspunkt i fysiske eksperimenter med LEGOs RCX [41, 58], som er beskrevet i afsnit 5.3.4.

Begge understøtter med eksperimenter, at IR kommunikation er ganske følsom over for ydre påvirkninger som lys og udformningen af rummet, de befinder sig i. Samtidigt påvises det, at rækkevidden for IR bølger er forholdsvist begrænset.

Sachs et al. viser, at antallet af IR pakker, der modtages af en RCX falder markant hvis modtageren udsættes for kraftigt lys fra en halogenpære. Desuden er der en tendens til at jo større pakker, der sendes, jo færre pakker når frem.

I Støys eksperimentelle opstilling står en række RCXer med IR transmittere og recievere pegende opad. Eksperimentet viser blandt andet, at jo længere modtageren er fra senderen jo færre pakker når frem og at hvis afstanden til loftet øges, hvilket betyder at afstanden som en IR pakke skal tilbagelægge mellem sender og modtager bliver længere, jo færre pakker når frem.

Fordele ved IR kommunikation:

- Billig teknologi.
- Mulighed for retningsbestemt kommunikation.

Ulemper ved IR kommunikation:

- Ingen mulighed for kommunikation med enheder i andre rum.
- Bliver let forstyrret af ydre påvirkninger som lys.
- Kan kun sjældent kommunikere med enheder i alle retninger.
- Modtageren kan miste beskeder fra senderen, hvis modtageren befinder sig i en blind vinkel i forhold til senderen.

5.3 Robotprodukter

I det følgende vil en række kommercielle robotprodukter bliver præsenteret for at undersøge, om der findes robotter, der er mere velegnede til at implementere robotspil end Spybots.

En fælles egenskab for de valgte robotter er, at de kan kommunikere. I præsentationen vil der være være en kort gennemgang af hver enkelt robot, hvor mulighederne for in- og output vil blive opridset herunder robotternes kommunikationsmuligheder. Prisen vil også blive nævnt, da man bør holde sig for øje at robotspil potentiet kunne være fremtidens julegaver og dermed spiller prisen også en rolle.

5.3.1 Khepera II

Khepera II, der er illustreret i figur 5.3, er en forbedret version af Khepera, der benyttes i forskningssammenhænge på en række universiteter. Grundrobotten er udstyret med otte infrarøde sensorer, der blandt andet kan benyttes til at bedømme afstand til objekter. Desuden er der omdrejningstæller på hjulene.

Det er muligt at købe ekstraudstyr til Khepera II robotten, blandt andet et radiokommunikationsmodul, der gør det muligt for en Khepera at sende beskeder til andre robotter eller til en PC. Der kan kommunikeres på to forskellige frekvenser $418MHz$ eller $433,920MHz$. Rækkevidden er ti meter og der kan sendes med en hastighed på op til 9600 baud; dog oplyser producenten, at 4800 baud er det typiske. Den software, der leveres med Khepera II understøtter radiokommunikationen for op til 32 robotter på en gang.

Af andet ekstraudstyr til Khepera II robotten kan nævnes kamera, gribeklo samt lineært eller matrix syn.

Pris for en Khepera er omkring 21.000 kr. ved køb af 10 stk. og med universitetsrabat. Tillæg for radiokommunikationsudstyr og kamera ca. 14.000. Oplysningerne er stammer fra [8, 34].

FIGUR 5.3: Khepera II

5.3.2 CyberMaster

CyberMaster fra LEGO (figur 5.4) er som LEGO's andre produkter fleksibelt med hensyn til det fysiske design. CyberMaster skal programmeres via en PC. Programmet kan enten downloades til robotten eller den kan styres fra et program der bliver afviklet på en PC.

FIGUR 5.4: LEGOs CyberMaster

Den kan modtage input via trykfølere, der tilknyttes inputportene. Med CyberMaster sættet følger trykfølere med tre forskellige modstande. Den har desuden mulighed for at modtage radio beskeder.

Cybermaster har to integrerede motorer samt en ekstra port til ekstra aktuator og kan desuden kommunikere via radiobølger. Den har desuden en indbygget højtaler. CyberMaster kan programmeres i et grafiske programmeringsmiljø eller

i *Not Quite C* [14].

CyberMaster men koster omkring 1200 kr.

5.3.3 AIBO

AIBO er et meget avanceret og dyrt stykke legetøj, se figur 5.5. Det nuttede væsen er udstyret med en 64 bits processor, og har mulighed for at kommunikere via trådløst LAN. Programmer til AIBO kan skrives i en variant af C++ som bliver overført til robotten via en memory stick [5]. AIBO benyttes på flere universiteter i Europa og USA [5].

FIGUR 5.5: Sonys AIBO robot

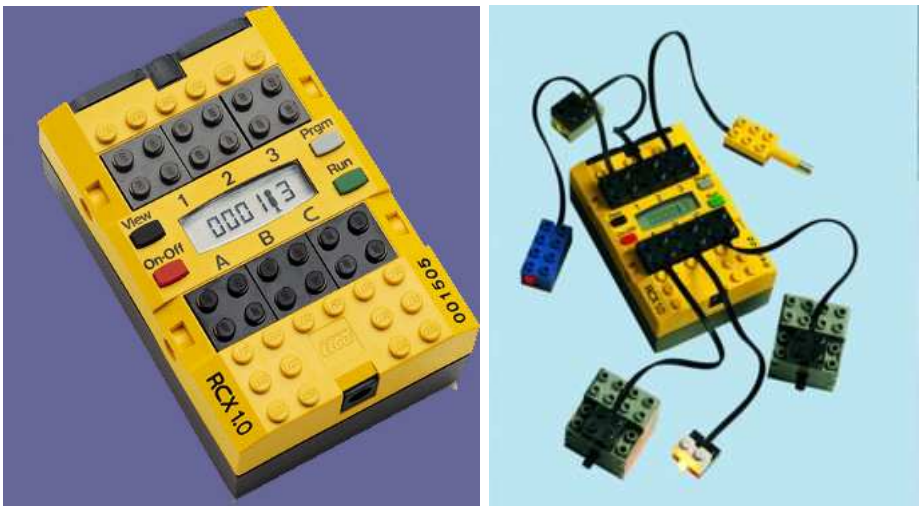
AIBO har mulighed for input fra 12 trykfølere, der blandt andet befinder sig oven på hovedet, på snuden, på benene og forrest og bagerst på ryggen. Den er desuden udstyret med farvekamera (350,000 pixels), stereo mikrofon, temperaturmåler, accelerationssensor og en infrarød afstandsmåler. AIBO har mulighed for at modtage beskeder via radiokommunikation [5].

Af aktuatorer har AIBO 18 bevæglige led. To i hvert ben, i hoved, mund, øre og hale. Hvilket skulle gøre den i stand til at præstere 250 forskellige bevægelser. Desuden har den 24 lysdioder fordelt i hoved (øjne) og hale, stereo højtalere og har mulighed for trådløs radiokommunikation.

Pris 15.400 kr. men der gives rabat til uddannelsesinstitutioner.

5.3.4 LEGO MindStorms RCX

RCX'en er den mest udbredte af LEGOs programmerbare klodser og er blevet til genstand for adskillige undersøgelser, der har dannet grundlag for en række artikler, specialer og ph.d. afhandlinger; især inden for områderne datalogi og ingeniørvidenskab. RCX'en ses i figur 5.6.



FIGUR 5.6: LEGOs RCX med og uden sensorer og aktuatorer

RCX'en kan modtage input fra fire trykknapper, en infrarød modtager, timere, batteriets strømniveau og sidst men ikke mindst tre inputporte, der kan forbindes med en række sensorer. Af sensorer kan nævnes trykføler, lysføler, omdrejningstæller og temperaturmåler.

Af outputenheder har RCX'en: en infrarød transmitter, LCD display, indbygget højtaler og tre outputporte. Til hver outputport kan der forbindes forskellige aktuatorer som motorer, lyspærer og lydenheder, der kan afgive biplyde.

RCX'en sender stabilt IR beskeder med en hastighed på 2400 baud [20] og har mulighed for at sende og modtage VLL (Virtual Light Link) koder, når firmwaren til RIS 2.0 er installeret.

Kontrolprogrammer til RCX'en kan blandt andet programmeres i det grafiske programmeringsmiljø RIS (Robotics Invention System), skrives i højniveau sproget C eller kodes som assemblykode. Et kontrolprogram overføres til RCX'en fra en infrarød sendeenhed, kaldet et tårn, tilkoblet en arbejdsstation til RCX'ens infrarøde modtagerenhed.

Pris ca. 2000 kr. En del sensorer og aktuatorer medfølger, samt en del LEGO klodser der kan benyttes til konstruktion af sin egen robot [12].

5.3.5 Spybot

LEGOs Spybotics sæt er LEGOs produkter i rækken af programmerbare klodser. En Spybot med fjernbetjening er illustreret i figur 5.7. Spybotproduktet adskiller sig fra de øvrige programmerbare enheder, som er beskrevet i dette speciale ved at minde mere om et spil end et egentligt programmeringslegetøj. Desuden er fokus flyttet fra leg-læring til underholdning.
Spybottens kan modtage input fra to trykknapper, en indbygget tryksensor og

FIGUR 5.7: LEGO Spybot med fjernbetjening.

en indbygget lysføler. Desuden kan Spybotten modtage IR beskeder. Af outputmuligheder har Spybotten to motorer, 8 lysdioder, en indbygget højtaler (piezo) og kan udsende IR og VLL beskeder.

Spybotten er udstyret med 2 IR modtagere og 4 IR transmittere og det gør den i stand til at afgøre, hvor langt andre Spybots og fjernbetjeninger befinder sig fra den inden for de tre zoner *here, there* og *anywhere*. Desuden kan den afgøre, i hvilken retning andre Spybots og fjernbetjeninger befinder sig. Forhold omkring Spybottens brug til bedømmelse af afstande mv uddybes i kapitel 6.

Spybotten leveres med fjernbetjening, en CDROM med byggevejledninger samt Spybotics spillet. Spybotics spillet giver mulighed for ganske simpel programmering af agenter. Der kan altså foretages en række valg der udmønter sig i en given adfærd. For de mere avancerede brugere kan Spybotten programmeres i *MindScript* eller *LASM* [1] og desuden er der lavet en *Not quite C* variant til LEGO Spybotics [14].

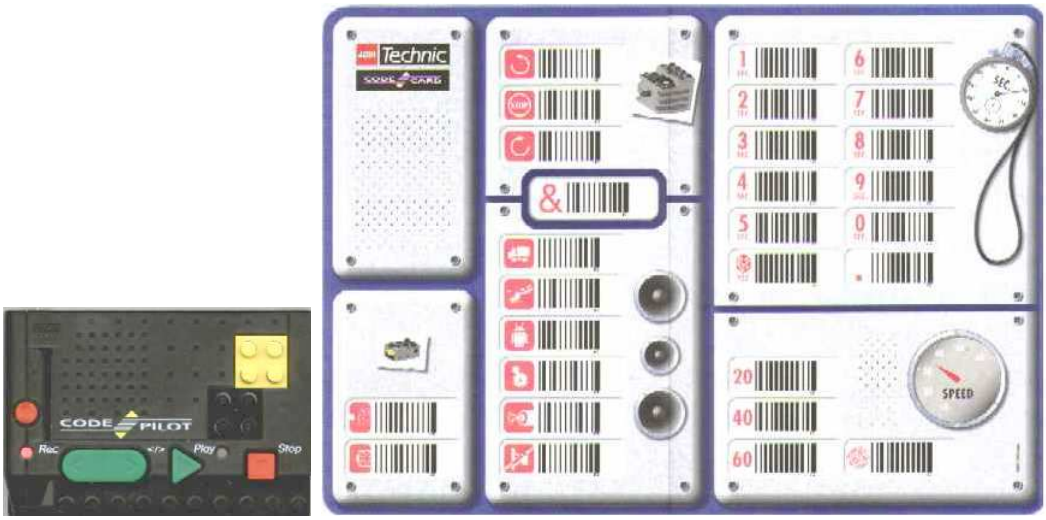
Prisen for et LEGO Spybotics sæt er 5-800 kr [12].

5.3.6 CodePilot

LEGOs CodePilot, som er en af LEGOs første programmerbare enheder, kan ses på figur 5.8.

Den har en indgang der kan forbindes med en trykføler og en udgang, hvortil der kan kobles aktuatorer som motor, lyspære og lignende. Desuden har CodePilot en lille integreret højtaler. CodePilot programmeres via VLL koder. VLL er en 7 bits digital protokol, hvor lys kan sendes fra LEGOs USB tårn, Spybots eller Scouts til CodePilot. Den umiddelbart letteste måde at programmere CodePilot

[1] MindScript er LEGO's eget robotprogrammeringssprog og LASM er LEGO assemblysprog til robotprogrammering



FIGUR 5.8: LEGOs Code Pilot (tv.) med kodeark (th.)

på er ved hjælp af stregkoder. Der medfølger et ark med stregkode så brugeren kan opbygge et sekventielt program let og hurtigt, kodearket ses til højre på figur 5.8. Ulempen ved stregkodeprogrammeringen er, at det ikke er muligt at rette i koden. Så laver brugeren fejl i sit program, er denne nødt til at starte forfra.

CodePilot er udgået, men kostede omkring 800 kr. sammen med LEGO Technics lastbil. Den er medtaget da der er CodePilots til rådighed på Aarhus Universitet.

5.3.7 Scout

LEGOs Scout har samme dimensioner som RCX'en. Et billede af Scouten ses i figur 5.9.

Scouten har muligheden for input fra trykfølere, der sættes på de to inputporte og en integreret lysføler. Desuden har Scouten mulighed for at modtage VLL koder og infrarøde beskeder. I forbindelse med lyssensoren er der en lysdiode, der lyser, hvis lysforholdene ændrer sig. Ligesom på RCX'en er der fire trykknapper, der benyttes til at tænde og slukke for enheden, vælge programmer osv.

Der kan påsættes aktuatorer på to de sorte porte og der kan afsendes VLL koder. Desuden har Scouten en integreret højtaler. I forbindelse med hver output port er der to pileformede lysdioder, der indikerer en påsat motors kørselsretning. Midt på enheden er der et LCD display.

Hvor RCX'en kræver en PC for at kunne programmeres, kan Scouten programmeres direkte på enheden ved at foretage en række valg ved hjælp af den grå og sorte trykknap. Herved defineres hvordan Scouten skal reagere på tryk og lys samt hvilket lyden, den skal benytte. Brugerens program kan overskues på LCD displayet midt på enheden.

FIGUR 5.9: LEGOs Scout

Adfærden for et typisk Scout program kunne være kør fremad i zigzak, hvis trykfølerne påvirkes, så kør baglæns og drej, hvis der er lyskilder, i nærheden så undgå dem.

Programmeringsmulighederne direkte på Scouten er begrænsede, men der er mulighed for at skrive et program i fx MindScript eller LASM og downloade det til scouten via et IR tårn og hermed er der mulighed for at skrive mere komplicerede programmer.

Prisen er omkring 600 kr. inklusiv klodser sensorer og aktuatorer.

5.3.8 MicroScout

LEGOs MicroScouts inputmuligheder er tre trykknapper, henholdsvis tænd/sluk, skift program og start/stop program, samt en indbygget lyssensor. MicroScouten kan ses på figur 5.10. Af aktuatore, har den et encifret LCD display der viser hvilket program brugeren, har valgt (1-7 eller P), en indbygget motor og en indbygget højtaler, der kan sige biplyde.

MicroScouten har syv indbyggede programmer fx et program, der udløser en alarm, hvis der sker ændringer i omgivelsernes lysforhold efter programmet er startet, og et hvor motoren kører fem sekunder den ene vej, derefter fem sekunder den anden vej og gentager dette 3 gange.

Et MicroScout program består af op til 15 instruktioner og enheden afspiller altid en biplyd, inden en motorinstruktion afvikles. MicroScouten programmeres vha VLL koder, der kan sendes til den fra USB tårn, Scout, Spybot eller lignende.

Prisen for en MicroScout er omkring 500 kr. sammen med en R2D2 Star Wars robot.

FIGUR 5.10: LEGOs MicroScout

5.4 Diskussion af robotvalg

Af de otte robotprodukter, der blev nævnt ovenfor kan de fem første indgå som autonome modstandere i robotspil. De tre sidste er for simple til at indgå i spil som modstander men vil kunne indgå i robotspil som aktive forhindringer, som kunne være en del af banen.

De to robotprodukter, der umiddelbart er mest tiltalende er AIBO robotten og Khepera II med kamera og radiokommunikationsudstyr. Det skyldes, at disse to robotter har en bred vifte af inputmuligheder og derfor vil kunne indgå i mange typer robotspil.

Desværre er prisen forholdsvis høj, så hvis et spil skal indeholde seks robotter, bliver den samlede pris omkring 100.000 kr. eller mere, hvilket uden tvivl vil forhindre, at spillet bliver udbredt.

Når sandheden skal frem, så er de robotprodukter der er tilgængelige på Aarhus Universitet LEGO's RCX og LEGO's Spybot. De har begge har en fordel ved deres fleksible design. Når valget står mellem disse to produkter, er ingen af produkterne umiddelbart mest oplagte.

LEGOs RCX er langt bedre dokumenteret end Spybotten, blandt andet fordi den er ældre, mere udbredt og har indgået i en lang række artikler fra universiteter i både Europa og USA. Dermed vil det være langt lettere at få hjælp til at løse hardware og software problemer fra entusiaster over hele verden. RCX'en har desuden en bredere vifte af in- og output muligheder.

Omvendt er Spybotten ideel, idet den lægger op til at kunne indgå i spil. Den er udstyret med 4 IR sendere og 2 IR modtagere, som gør det muligt for Spybotten at bedømme afstande til andre Spybots samt afgøre i hvilken retning de befinder sig i; en navigationsmulighed som RCX'en ikke tilbyder. Desuden hører der fjernbetjeninger med til hver Spybot, som giver op til tre spillere af

gangen let mulighed for at kontrollere hver deres aktant, hvilket danner grundlag for et godt gameplay.

På baggrund af de overvejelser, samt de erfaringer jeg har med både RCX'en fra kurset *Embedded Systems Embodied Agents* og med Spybotten falder valget af robot til en robotspilsprototype på Spybotten.

Når LEGO produkter er valgt som aktører i et robotspil og når der samtidig er mulighed for at benytte LEGO til aktive elementer på banerne, er det kun naturligt, at banerne i den robotspilsprototype der vil blive konstrueret, også består helt eller delvist af LEGO. LEGO signalerer *leg*[2] og derfor kan et naturligt element i et robotspil meget vel være processen med at konstruere banerne. Dette aspekt vil blive undersøgt nærmere i forbindelse med afprøvning af robotspilsprototypen sammen med børn.

5.5 Spybotics produktet

LEGO Spybots produktet består af en CD-ROM med software og en række klodser til at konstruere en Spybot samt en fjernbetjening til at styre den med. For at kunne benytte sig af produktet skal man være i besiddelse af en computer med Windows styresystem. Historien i Spybotics er, at du er en hemmelig agent og Spybotten er din trofaste makker. Sammen skal I løse opgaver rundt omkring i verden. Der er ti forskellige opgaver/missioner i alt, samt muligheden for at konstruere sine egne missioner. Når en mission vælges, briefes spilleren om baggrunden for missionen. Det sker ved hjælp af flotte grafiske animerede filmsekvenser. En mission, kan være at man skal styre sin Spybot rundt i et atomkraftværk og forsegle lækager, så det ikke fører til en global miljøkatastrofe. Efter briefingen kan enkelte funktioner programmeres ind i Spybotten og det kan bestemmes, hvordan den skal reagere på knaptryk på den tilhørende fjernbetjening. Spybotten kan fx programmeres til automatisk at skyde, hvis der kommer en anden Spybot i nærheden af den og man kan vælge, at Spybotten skal have et beskyttelsesskjold, hvis en bestemt knap på fjernbetjeningen påvirkes. Der er desuden mulighed for at programmere en autonom robot med et simpelt kontrolprogram.

Herefter downloades programmet til Spybotten. Selve spillet eller løsningen af missionen finder sted på et gulv; atomreaktorer, lækager med videre overlades til spillerens fantasi. Spillet går ud på at navigere Spybotten rundt og løse opgaver i en fantasiverden.

5.6 Første spilscenarie

Min første grundigere undersøgelse af produktet var noget skuffende, for de missioner der medfulgte robotten var meget simple og krævede at spilleren eksempeltvist forestillede sig at gulvet i børne- eller arbejdsværelset var en nedsmeltet

[2]LEGO er en forkortelse af ordene "leg godt"

atomkraftreaktor med dødsensfarlige lækager som skulle forsegles med Spybottens "laserkanon". Ideen er såmænd udemærket og ville nok have virket, hvis der på den ene eller anden måde var tilføjet elementer så spilleren rent faktisk havde mulighed for at se reaktorelementer i sine omgivelser. Men i spillets nuværende form er missionerne faktisk både kedelige og frustrerende. Kedelige på grund af banernes simpelhed og frustrerende, fordi ens Spybot til tider reagerer så dårligt på ens tryk på fjernbetjeningen, at spilleren ikke har indtryk af at have ordentlig kontrol over sin Spybot. Man har svært ved at slippe fornemmelsen af, at man bare sidder alene på sit værelse og kører rundt med en fjernstyret bil med mangelfulde styreegenskaber.

Et element ved Spybotprogrammeringsmiljøet viste sig dog at være ganske morsomt; "gør det selv missionen" hvor spilleren selv har mulighed for at programmere sin Spybot og dens modstandere. Her skal spilleren selv finde på en mission, samt programmere både agenten og dens modstandere. Selve processen med at programmere minder om den konstruktionsleg, børn oplever når de bygger med LEGO, men det spil man når frem til mangler stadig fysiske omgivelser. Derfor opstod ideen med at konstruere en form for computerspil, hvor banerne var opbygget af LEGO klodser. Spillet var inspireret af actionskydespil som *Doom*, *Quake*, *Wolfenstein 3D*, der alle er spil hvor spilleren bevæger aktanten rundt i et 3D miljø og oplever verden fra dennes synsvinkel. Spillene går i alt sin enkelthed ud på at skyde på alt og alle, samt løse simple opgaver for at kunne komme videre. Fx kan en opgave være at trykke på to knapper i en bestemt rækkefølge for at åbne porten til næste bane. Spilbeskrivelser af de nævnet computerspil findes i appendix A side 119. Christian Ulrik Andersen, ph.d.i computerspil ved institut for informations- og medievidenskab, arbejdede samtidigt med Spybotics, så vi slog pjalterne sammen og fik i fællesskab konstrueret et spil.

Det konstruerede spil blev en udemærket prototype. Det bestod af autonome modstander Spybots samt en række stillestående forhindringer. Der var i alt fire forhindringer:

1. En vippeplade der var placeret hen over en mur (figur 5.11). Her skulle spilleren styre Spybotten over pladen - det er faktisk sværere end det lyder.

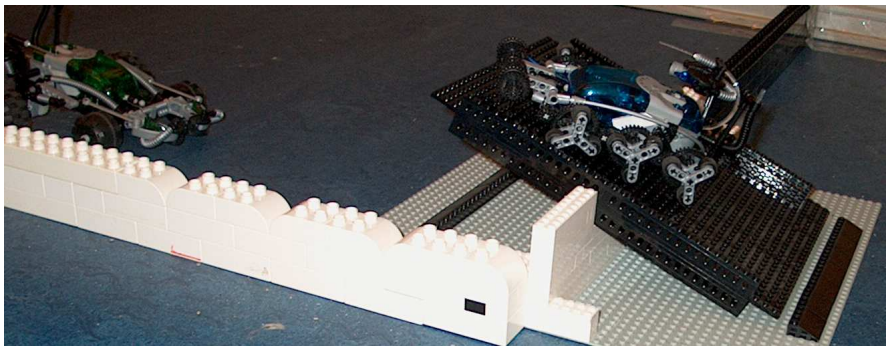
2. En mur med en rød knap på (figur 5.12). Når der trykkes på knappen åbnes porten i muren i 7 sekunder. Døren og knappen blev styret af en LEGO CodePilot[3]. For at gøre denne forhindring sværere blev 1-2 autonome Spybots indsat for at være i vejen for spilleren.

3. Den mest komplicerede forhindring bestod af et skråplan der virkede som kontakt. Denne skulle spilleren køre sin Spybot op ad og herefter kunne

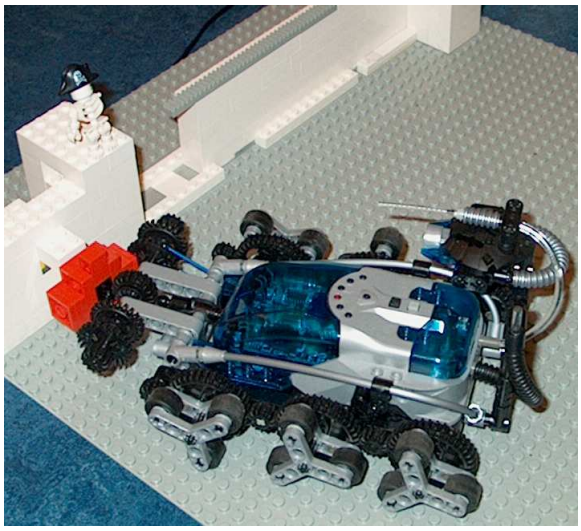
[3]LEGO CodePilot er en simpel programmerbar enhed der kan påsættes en motor og en trykknap. Enheden programmeres vha. stregkoder. En nærmere gennemgang findes i afsnit 5.3.6 på side 50

en anden vippekontakt udløses så en dør blev åbnet og spilleren kunne fortsætte.

4. Sidste forhindring bestod af tre håndtag som hverisær skulle vippes til en bestemt side for at åbne et stort faldgitter der spærrede vejen til målet (figur 5.13). Igen blev der indsat et par autonome robotter, der skulle gøre det svære for spillerne at komme forbi denne forhindring.

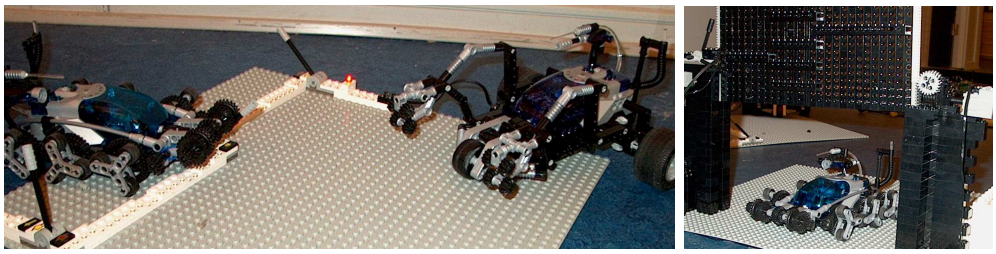


FIGUR 5.11: Spybot på vej over vippepladen. Modstander står på lur.



FIGUR 5.12: Spillerens Spybot åbner dør ved at trykke på knap.

Dette simple spil er ud over at være blevet selvtestet og testet på legesyge medstuderende også blevet afprøvet af både børn og voksne ved tre forskellige lejligheder. Først ved kulturnatten i Århus 2002, derefter ved NordiCHI konferencen i efteråret 2002. Ved begge disse lejligheder var det både Christian og



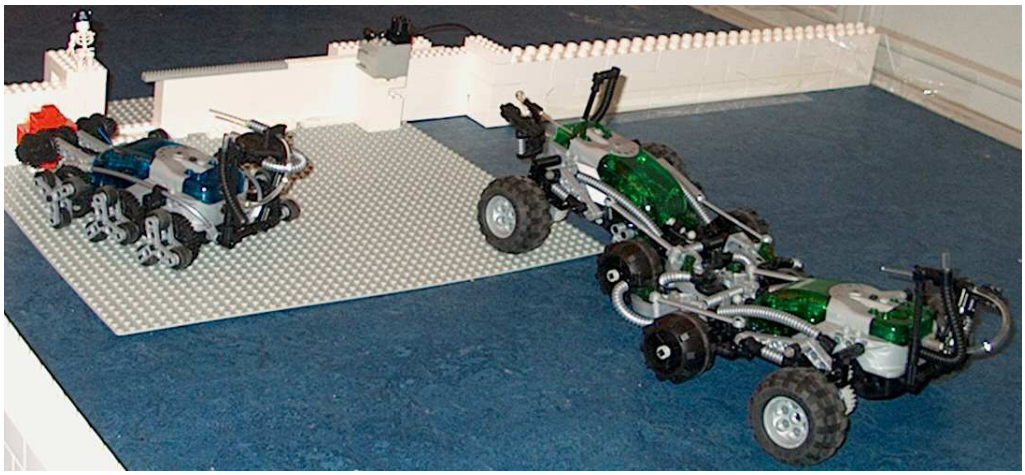
FIGUR 5.13: Sidste forhindring. Tv. Spillerens Spybot vipper håndtag, mens den bekæmpes af modstander. Th. Spybot kører gennem porten, der er sidste forhindring.

undertegnede der instruerede spillerne hvordan spillet skal spilles og observerede, hvorledes spillets forløb foregik. Spillet blev desuden afprøvet i december på Huset i Århus.

På trods af spillets enkelthed var det både krævende og morsomt at gennemføre. Den gennemsnitlige spiller var omkring 5-6 minutter om at gennemføre banen og de fleste måtte have nogle hints undervejs. Ved kulturnatsarrangementet var vi så heldige at få afprøvet vores spil på børn i 9-10 års alderen, hvilket er en del af målgruppen for Spybots. Det viste sig også, at de klarede sig lige så godt eller bedre end voksne spillere. Ialt nåede omkring femten børn og tredive-fyrre voksne at prøve spillet.

De fleste spillere mente at Spybotspillet var et interessant og morsomt supplement eller alternativ til de allerede eksisterende computerspil. Spillerne syntes generelt godt om den udfordring, det er at styre en fysisk robot rundt på banerne. Det var morsomt at kunne beskyde de fjendtlige Spybots de første par gange, man stødte på dem, selv om de ikke kunne nedkæmpes. Ligeledes fandt flere det sjovt at spillerens robot kunne køre fast i de fysiske forhindringer, hvis spilleren ikke styrede den ordenligt. Disse fastkøringer kunne som oftes løses ved behændigt fjernbetjeningsarbejde.

Der var et kritikpunkt der gik igen fra både børn og voksne ved begge arrangementer; de autonome robotter var for uintelligente til at de var specielt interessante at kæmpe imod. Hvis de autonome Spybots optrådte i par, brugte de lige så mange kræfter på at bekæmpe hinanden som på at bekæmpe modstanderen. Dette er illustreret i figur 5.14 hvor man kan se to modstandere i indbyrdes kamp, mens spilleren bevæger sig imod den port de burde beskytte. De kunne altså på ingen måde arbejde sammen eller tilpasse sig spillerens strategi. Desuden havde de autonome Spybots ingen mulighed for at nedkæmpe spilleren eller visa versa, på trods af at både menneskestyrede og autonome Spybots kunne skyde med "laser" og ramme hinanden, så de rystede.



FIGUR 5.14: Spiller der kører forbi to modstander der bekæmper hinanden.

5.7 Spybots i 0.B

Ud over afprøvningen af det konstruerede spil blev Spybotterne afprøvet i børnehaveklassen 0.B. på Katrinebjerskolen i februar 2003 i forbindelse med en temauge om robotter, arrangeret af Klaus Testrup underviser i drama på pædagogseminariet i Randers og Ole Caprani lektor ved Aarhus universitet.

Måden Spybots blev benyttet i 0.B. var noget anderledes end ved de tidligere arrangementer. Her var det ikke Spybots, der var det primære, men robotter generelt. Børnehaveklassebørnene var alle omkring seks år gamle - altså tre år yngre end målgruppen for LEGO Spybots. Så Spybotforløbet med børnene fik derfor mere karakter af leg end spil.

Spybotlegene i børnehaveklassen foregik i mindre grupper bestående af fire børn - to drenge og to piger. Legene opstod alle spontant som samspil mellem Klaus, børnene og jeg. De første lege bestod af at lære at navigere med de Spybots, som børnene selv kunne styre. Derpå blev der blandt andet leget "robot tagfat" og kapløb. De autonome Spybots blev også benyttet i lege, hvor to baser var placeret i hver sin ende af lokalet. Midt i lokalet stod en række autonome robotter mellem en masse klodser og andre forhindringer. Legen bestod i at styre sin Spybot forbi midten uden at vælte kloder og uden at blive forfulgt og skudt af de autonome Spybots.

Det interessante ved sidstnævnte leg er at den indeholder elementer der minder om det spilscenarie som blev konstrueret af Christian og jeg. Forhindringerne var anderledes men overordnet set havde både vores Spybotspil og børnenes Spybotleg et startsted, nogle forhindringer og et målområde.

Erfaringerne med LEGO Spybotics har vist, at der er nogle helt fundamentale mangler ved Spybotics spillet på den måde som de enkelte missioner er udformet. Dog er der mulighed for at konstruere ganske underholdende spil ved hjælp at

nogle Spybots, Spybotics "gør det selv" programmer og en bunke LEGO klodser eller andre forhindringer. "Gør det selv" programmerne har dog det problem, at de autonome Spybots, det er muligt konstruere er alt for simple. Der er ingen mulighed for at skelne ven fra fjende og samarbejde med andre autonome Spybots er ikke muligt. Der skal dog herske tvivl om, at LEGO Spybotics alligevel er et rigtigt godt produkt til prisen.

Del III
Implementering og afprøvning

Efter at LEGOs Spybots er blevet valgt som hardwareplatform til de autonome robotter, der skal være aktører i det endelige spil, er det nyttigt at foretage en række tests af robotterne for at afdække egenskaber, der er relevante til spilsammenhæng. Det er især vigtigt at undersøge, hvordan IR kommunikationen mellem robotterne fungerer under forskellige forhold, da IR beskeder er Spybots største kilde til information omkring deres omgivelser. Efter undersøgelserne skal der skabes en arkitektur for kontrolprogrammet til spillets aktører. Kontrolprogrammet skal implementeres og slutteligt skal spillet afprøves.

6 Undersøgelser af Spybots

Inden kontrolprogrammet for det konkrete spil kan designes og implementeres er det som programmør vigtigt at være bekendt med muligheder og begrænsninger for den hardwareplatform, som programmet skal afvikles på. Derfor skal der indsamles viden om LEGO Spybots platformen. En del af den fornødne viden har vi allerede i den dokumentation, der blev beskrevet i 5.3.5, men der er en række aspekter ved Spybotterne, som er relevante i forbindelse med spil, som ikke er dokumenteret, ihvertfald ikke i dokumenter, der er offentligt tilgængelige.

6.1 Pakkeformater og sendehyppigheder

Spybotten sender pakker med en hastighed på 4800 bits/sekund. Spybotten kan sende flere typer beskeder. To af beskedtyperne benyttes til kommunikation mellem Spybots og det er henholdsvis **pingpakker** på 4 bytes og **datapakker** på 7 bytes. Hver byte fortolkes som et positivt heltal i intervallet 0-255. Pingpakkerne, der udsendes med et regelmæssigt interval for at gøre omkringværende Spybots opmærksomme på dens tilstedeværelse. Dette interval er som udgangspunkt fire gange i sekundet. I pingpakken kan programmøren, hvis denne skriver i MindScript, frit benytte en ud af de fire bytes. Datapakken benyttes til udveksling af data mellem Spybots og giver programmøren mulighed for frit at benytte 3 bytes. Formatet af ping- og datapakkerne, når de bliver afsendt, kan ses i figur 6.1 og 6.2.

6.1.1 Pingpakkens opbygning

Figur 6.1 viser opbygningen af en pingpakke. Formatet er dels fundet ved empiriske undersøgelser, hvor pingpakker blev sendt til en PC via et IR tårn og dels fra manualen til Spybotics [61].

Header	BotID	User	Checksum

FIGUR 6.1: Opbygning af 4 bytes pingpakke

Header Headeren består af to dele på 4 bits hver. Skrevet i binære tal består første del af de fire bits 1000. Anden del er et LinkID, som angiver hvilken fjernbetjening der kan kommunikere med denne Spybot. LinkID kan være 0-3, hvor 0 angiver, at Spybotten ikke er knyttet til nogen fjernbetjening og 1-3 angiver fjernbetjeningskanal.

Dermed kan der højest være tre forskellige spillere med fjernbetjeninger i et spil. I binære tal er headeren altså tallene 10000000-10000011, hvilket er en let genkendelig header. Programmøren af en Spybot har ingen kontrol

over headeren, bortset fra at han kan knytte en Spybot til en bestemt fjernbetjening.

BotID Er et unikt ID på den Spybot, der udsender pakken. Har to Spybots det samme ID, hvilket forekommer ofte, da alle Spybots som udgangspunkt har tallet 8 som ID, så vælger de hver især et nyt ID i intervallet 8-255. Programmøren har ikke indflydelse på en Spybots ID.

User Byte der bruges af programmøren.

Checksum Checksummen beregnes som følger:

$$Checksum = 128 - (LinkID + BotID + User).$$

6.1.2 Datapakkens opbygning

Figur 6.2 viser opbygningen af en datapakke. Formatet er som ovenfor dels fundet ved empiriske undersøgelser og dels fra manualen til Spybotics [61].

Header	BotID	TargetID	User1	User2	User3	Checksum

FIGUR 6.2: Opbygning af 7 bytes datapakke

Header Headeren består som ovenfor af to dele. Første del er 1000, som i pingpakkerne. De næste 4 bits angiver Spybottens **SendMode** som angiver måden, hvorpå Spybotten sender sine beskeder. Fx angiver headeren 10001000 at Spybotten udelukkende sender til sit mål eller **target** og 10001010 angiver, at beskeden bliver sendt til alle Spybots.

BotID Som ovenfor.

TargetID Angiver BotID for den robot, der er denne Spybots mål (target). Målet vil ofte være den modstanderrobot, der er tættest på. Har robotten ingen modstander er denne byte 0000 0001.

User1-3 Som ovenfor.

Checksum Checksummen beregnes ud fra formlen:

$$Checksum = 128 - (Kanal + BotID + TargetID + User + User1 + User2 + User3).$$

Det er også muligt at sende beskeder til, og modtage beskeder fra en RCX eller en Scout. Disse beskeder er ni bytes lange, selve beskeden består dog kun af en byte, de resterende otte bytes er header og checksum. Disse pakker bliver sendt med en hastighed på 2400 baud [61].

6.1.3 Spybot fjernbetjeningsbeskeder

FIGUR 6.3: Fjernbetjeningen fra Spybotsættet

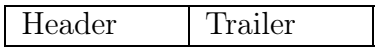
Fjernbetjeninger, der medfølger Spybot sættet, se figur 6.3, udsender IR beskeder, som er forskellige, alt efter hvilken af de tre kanalnumre fjernbetjeningen er tilknyttet. Fjernbetjeningen har tre indstillingsmuligheder, kaldet **modes**, og valget af mode har ligeledes betydning for typen af beskeder, der udsendes. De tre modes er **control mode**, **link mode** og **action mode**, som indstilles ved at dreje den lille grå tap nederst på fjernbetjeningen. Control mode er til venstre, link mode i midten og action mode til højre.

Control mode Her udsender fjernbetjeningen kun IR beskeder, når knapperne påvirkes. Denne indstilling benyttes til at styre Spybotten. De to grå knapper til højre, benyttes til at styre Spybottens højre motor, og de to knapper til venstre styrer den venstre motor, når man ser Spybotten forfra. Da man ofte står bag ved Spybotten og styrer den, kræver styringen en smule øvelse.

Link mode Benyttes til at vælge, hvilken af de tre kanaler, fjernbetjeningen skal kommunikere via. Det gøres ved at trykke en af knapperne 1-3 (de tre øverste) og herefter knappen i højre hjørne. Holdes fjernbetjeningen tæt på Spybotten, når der vælges kanal, får Spybotten samme kanal (LinkID) som fjernbetjeningen. I link mode udsendes der kun besked, når knappen nederst til højre påvirkes. Så udsendes der besked for at give Spybots tæt på besked om at linke til denne fjernbetjeningskanal.

Action mode Her udsender fjernbetjeningen fire pingpakker i sekundet som adskiller sig fra Spybotternes beskeder. Når der trykkes på knapperne udsen-

des der beskeder der adskiller sig fra beskederne i control mode. De beskeder som fjernbetjeningerne udsender i action mode er illustreret i figur 6.4.



FIGUR 6.4: Format af fjernbetjeningsbeskeder samt pingpakker i action mode

Header Headeren består af to dele på 4 bits hver. Skrevet i binære tal er første del 1001. 1001 indikerer, at beskeden er fra en fjernbetjening. Headerens anden del er LinkID, der som ovenfor angiver hvilken, af kanalerne (1-3) fjernbetjeninger er tilknyttet.

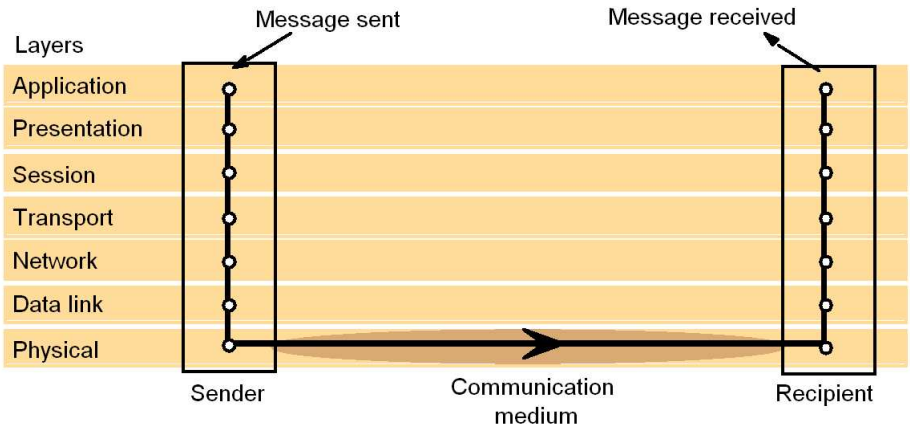
Trailer Traileren består dels af en knaptryksværdi på 4 bits og dels af en checksum på 4 bits. Knaptryksværdien angiver hvilken af knapperne 1-5 der er trykket på. Pingpakkerne fra fjernbetjeningen og de beskeder, der udsendes, når man trykker på en knap, har samme format bortset fra, at knaptryksværdien er 0000 i pingbeskederne. Pakkerne, der sendes i link mode har også samme format, men her er knaptryksværdien 1111. Checksummen beregnes ud fra formlen:

$$Checksum = 16 - (LinkID + Tryk).$$

6.2 Spybottens kommunikationsprotokol

En kommunikationsprotokol er en standard der gør det muligt at udveksle data. Protokollen gør det muligt at give den oprindelige besked et format, som kan sendes trådløst eller via et kabel og blive modtaget af den eller de rigtige modtagere. For at det kan lade sig gøre skal en besked til tider deles op i mindre pakker inden afsendelse. Derfor bliver hver pakke, der skal sendes typisk påhæftet information, omkring modtager, afsender og lignende. På modtagersiden skal pakkerne efter modtagelse samles til besked igen og den ekstra information, der er blevet tilføjet til pakker og beskeder skal fjernes igen. En kommunikationsprotokol består af to stakke, en senderstak og en modtagerstak. Hver af disse stakke er delt op i flere lag og for hvert lag i senderstakken, tilføjes der information til besked eller pakker og i modtagerstakken fjernes denne ekstra information igen. I figur 6.5 er lagene i OSI (*Open Systems Interconnection*) protokolstakken illustreret [26].

Det er relevant at se på kommunikationsprotokollen i forbindelse med robotter i spil, idet en kommunikationsprotokol altid tilfører ekstra information til en besked, der skal sendes. Herved gøres beskeden større og dermed udnyttes båndbredden mindre optimalt.



FIGUR 6.5: Skematisk fremstilling af OSI protokollen. Lånt fra [26] s. 78.

Når vi har med Spybots at gøre, er kommunikationsprotokollen så godt som lukket land for alle andre end ansatte i LEGO, men det er muligt at komme med nogle kvalificerede bud på, hvordan den er opbygget.

Spybots beskeder er små (1-3 bytes) og de bliver ikke opdelt i mindre pakker, når de bliver afsendt. Derfor vil termen *pakker* blive benyttet om kommunikation mellem Spybots. Spybots kan som udgangspunkt kun kommunikere direkte, der er altså ikke tale om, at de kan fungere som router og sende pakker videre til andre. Dermed er de eneste lag fra OSI modellen, der bliver benyttet i Spybottens kommunikationsprotokol, *datalink laget* og *det fysiske lag*. Nedenfor er en beskrivelse af, hvad der formodentligt sker på de to lag i Spybottens kommunikationsprotokol.

Datalink laget Her tilføjes header, BotID og TargetID og pakketypen til pakken. Checksum udregnes og tilføjes på sendersiden. På modtagersiden benyttes header og checksummen til at frasortere støj og korrupte pakker og herefter fjernes header og checksum.

Det fysiske lag Det fysiske lag er selve hardwaren. Her bliver Spybotternes pakker sendt via en komponent, der hedder en UART [61] (*Universal Asynchronous Receiver Transmitter* [60] s. 193). En UART læser bytes fra databussen og sender dem bitvist via den infrarøde transmitter. UART'en tilføjer stop- og evt. paritetsbits til hver byte, inden de sendes, så UART'en på modtagersiden kan skelne pakkerne og identificere fejl i de enkelte bytes. I LEGOs RCX tilføjes der 3 ekstra bits pr. byte inden afsendelsen[1], og da

[1]I RCXen er der dog ikke tale om en UART, men en SCI (*Serial Communication Interface*) [29] section 11.

Spybotten og RCXen kan kommunikere er det nærliggende at antage, at UART'en i Spybotten ligeledes tilføjer 3 bits pr. byte.

Da der således sendes 11 bits pr. byte, og der maksimalt kan sendes 4800 bits i sekundet, kan der maximalt sendes:

$$\frac{4800\frac{bits}{sek}}{11\frac{bits}{byte}\cdot 4\frac{bytes}{pingpakke}} \approx 109\frac{pingpakker}{sek}$$

eller:

$$\frac{4800\frac{bits}{sek}}{11\frac{bits}{byte}\cdot 7\frac{bytes}{datapakke}} \approx 62\frac{datapakker}{sek}$$

Sagt med andre ord tager det i hvertfald 9 millisekunder at sende en pingpakke og i hvertfald 16 millisekunder at sende en datapakke.

6.3 Undersøgelse afsende- og modtagerforhold

I forbindelse med Spybots er en afdækning af aspekter omkring afsendelse og modtagelse af IR pakker vigtig, da de orienterer sig i forhold til hinanden ved hjælp af infrarøde pakker. Desuden er kommunikation ofte en vigtig forudsætning for samarbejde mellem robotter. I forhold til spil, hvor der indgår en række robotter, der skal kunne kommunikere for at kunne se hinanden, er det især interessant at undersøge hvad, der sker, når robotten skal håndtere så mange pakker, at den risikerer at miste nogle.

Er der implementeret en protokol der tager hånd om pakketab? Erfaringerne fra kurset *Computerspil som Æstetik/LEGO Spybotics* viste, at spillerne ofte havde problemer med at opnå fuld kontrol med Spybots, og at de autonome Spybots til tider havde problemer med at finde andre Spybots. Det kan være tegn på, at IR pakker går tabt i kommunikationen mellem fjernbetjeningen og Spybot og Spybots indbyrdes. Dermed er det sandsynligt, at der i LEGO Spybotics produktet ikke tages hånd om problemet med pakketab.

I det følgende er der gennemført en række undersøgelser for at afdække forskellige aspekter af kommunikationen på Spybots. Det er interessant at undersøge under hvilke forhold, der er størst tab af pakker, så pakketab i robotspil kan minimeres. Der er tre hovedårsager til, at en Spybot ikke modtager de IR pakker, som er sendt til den. I figur 6.6 er hovedårsagerne til pakketab opridset.

Kraftigt lys påvirker de infrarøde pakker markant, hvilket blandt andet dokumenters af Støy og Winfield et al. [58, 64]. Derfor er alle undersøgelser blevet udført under dæmpet belysning i et lokale uden tændte lysstofrør og sollys. Det er forhåbningen, at det herved er muligt at eliminere eller i det mindste reducere den første af de tre årsager fra figur 6.6.

Undersøgelserne er tilrettelagt således, at der vil blive set nærmere på de sidste to årsager til tab af pakker. Formålet er dermed primært at måle, hvornår

1. Der kan være IR støj i omgivelserne; fx sollys.
2. Pakkerne kan kollidere med IR pakker fra andre Spybots.
3. Antallet af pakker kan være så højt, at Spybotten ikke kan nå at håndtere dem alle.

FIGUR 6.6: De 3 hovedårsager til tab af IR pakker.

der sker pakketab forsaget af årsagerne nævnt i punkt 2 og 3.

I alt fem undersøgelser er blevet gennemført. Første undersøgelse skal afdække inden for hvilke afstande en Spybot kan se andre Spybots og bedømme afstanden til dem inden for de tre zoner *here, there* og *anywhere*. Undersøgelse nummer to og tre er en undersøgelse af, hvor mange pakker en Spybot kan håndtere uden at lide stort pakketab. Dermed undersøges punkt 3 i figur 6.6. I de to sidste undersøgelser afprøves det i hvor høj grad flere Spybots, der sender samtidigt, forstyrrer hinandens transmissioner. Her er det punkt 2 i figur 6.6, der ses nærmere på. For alle de udførte undersøgelser gælder, at de er blevet gennemført tre gange.

6.3.1 Abstrakt kode for undersøgelsesprogrammet

For at gennemføre de ønskede undersøgelser var det nødvendigt med henholdvist et sender- og et modtagerprogram. De benyttede programmer er skitseret som abstrakt kode i figur 6.7 og 6.8. Hver Spybot sender 500 eller 1000 pakker pr. undersøgelse og hver undersøgelse bliver gentaget midt tre gange. Årsagen til at gennemføre undersøgelse mindst tre gange i stedet for bare at sende tre gange så mange pakker, selv om i teorien burde give samme resultat, er, at det ikke nødvendigvis er ligegyldigt, hvornår kontrolprogrammerne bliver startet i forhold til hinanden. Men dette aspekt vil blive behandlet nedenfor.

Figur 6.7 og 6.8 illustrerer i abstrakt kode, hvordan henholdsvist sender- og modtagerrobotten er implementeret i undersøgelse 1 til 5. I senderprogrammet (figur 6.7) defineres en pause, der angiver, hvor mange tiendedel sekunder, der skal være mellem afsendelsen af hver pakke. Programmerer man Spybots i MindScript har man som programmør kun timere til rådighed med en opløsning på 100 millisekunder. Programmerne fungerer således, at senderen transmitterer et ID samt et af tallene fra 1 til 100 i kronologisk rækkefølge, med en hyppighed på mellem 1 og 10 pakker pr. sekund, afhængigt af værdi af pause. Når der senders mere end 100 pakker starter talrækken fra 1 igen.

Modtageren ved dermed hele tiden, hvilken pakke der bør være den næste, og får den en anden pakke end forventet, vides det, at der er foregået et pakketab. Har den sidst modtagne pakke fx indeholdt 8, bør den næste pakke indeholde tallet 9, men modtager man istedet en pakke med tallet 11, må der være gået to pakker tabt; nemlig pakkerne med henholdsvist 9 og 10. ID på den robot, der skal

```
start  program Sender
  timer t
  const pause
  const receiverID
  var   old_t
  var   msg←1
  start t
  old_t←t
  forever do
    if old_t + pause < t then
      old_t←t
      SendPackage(receiverID,msg)
      msg ++
    end if
    if msg = 101 then
      msg←1
    end if
  end while
end program
```

FIGUR 6.7: Program til afsendelse af pakker, skitseret som abstrakt kode.

modtage pakken medsendes, fordi det er nødvendigt i de to sidste undersøgelser hvor flere sendere transmitterer pakker til flere forskellige modtagere[2].

LEGO Spybots sender som udgangspunkt 4 pingpakker i sekundet under et spil. Derfor er det især interessant i de nedenstående undersøgelser at undersøge, hvor stort pakketab der er, når der sendes med netop denne hyppighed. Jeg har undersøgt sendehyppigheder på 5 og $3\frac{1}{3}$ pakker/sekund og det skyldes den ovennævnte timeropløsning på 100 millisekunder. Når LEGOs programmører kan sende fire pakker pr. sekund skyldes det, at de har programmeret i LEGOs assembly kode LASM, der stiller timere til rådighed med 10 millisekunders opløsning. Programmerne, der benyttes i undersøgelserne er ikke skrevet i *LASM*, dels fordi det er langt mere tidskrævende at skrive assembly kode[3], og dels fordi det endelige program alligevel bliver skrevet i MindScript.

Det skal bemærkes, at de pakker, der bliver sendt i undersøgelsen, består af syv bytes, hvor pingpakkerne kun består af fire bytes. De større datapakker er benyttet fordi, det ikke er muligt at ændre pingpakkernes brugerdefinerede del mellem hver afsendt pakke og dermed er det ikke muligt at afgøre, hvor mange

[2]Det skal for en ordens skyld bemærkes, at det ID jeg benytter, ikke er Spybotternes eget unikke ID, men et ID som jeg selv har defineret

[3]*Assembly language programming is difficult. Make no mistake about that. It is not for whimps and weaklings* [60] s. 485

```
start  program Receiver
  const myID
  const receiver_id
  var   msg_expected←1
  var   errors←0
  forever do
    if MessageReceived then
       if MessageReceived.receiverID = myID then
          if MessageReceived.msg <> msg_expected then
             if MessageReceived.msg > msg_expected then
               errors←(MessageReceived.msg-msg_expected)
             else
               errors←(MessageReceived.msg+100-msg_expected)
             end if
            msg_expected←MessageReceived.msg
          end if
         msg_expected++
       end if
    end if
    if msg_expected = 101 then
      msg_expected←1
    end if
  end while
end program
```

FIGUR 6.8: Program til modtagelse af pakker, skitseret i abstrakt kode.

pakker der tabes.

6.3.2 Undersøgelse 1

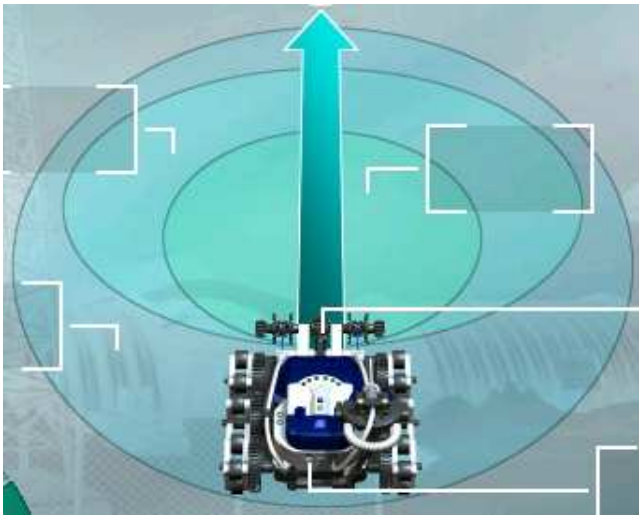
LEGOs Spybots er i stand til at afgøre, hvor langt andre Spybots og fjernbetjeninger til Spybotsættet er væk, samt hvad retning de befinder sig i. Det gøres ved hjælp af pingpakker, som hver enhed som udgangspunkt udsender fire gange i sekundet [61]. LEGO udnytter IR signalets fysiske egenskaber til at foretage afstandsbedømmelsen, men hvordan det præcist gøres er ikke offentligt tilgængeligt[4]. Afstandsbedømmelsen sker inden for tre zoner *here, there* og *anywhere*.

Spybots har til tider svært ved at finde hinanden, og årsagen til det kan være, at de ikke altid er i stand til at se hinanden. Det er min hypotese, at afstandene ikke er helt som i figur 6.9 og at grænserne mellem zonerne er udflydende. Målet

[4]At afstandsbedømmelsen foretages på baggrund af IR signalets fysiske karakter har jeg fået oplyst af Michael Barrett Andersen; medudvikler af LEGO Spybotics.

med den første undersøgelse er derfor at afgøre, hvor grænserne går mellem de forskellige zoner og dermed hvor Spybots kan se hinanden og hvor de ikke kan. Figur 6.9 viser, hvordan LEGO mener, de forskellig zoner er afgrænset. Som det ses er der ingen afstandsangivelse på illustrationen, så de vil blive kortlagt.

Undersøgelsen er gennemført med en stationær Spybot og en mobil Spybot. Den mobile Spybot blev placeret på forskellige afstande af, og i forskellige vinkler i forhold til den stationære Spybot. Den stationære Spybot angiver med sine lysdioder afstanden, der er til den mobile Spybot. Opstillingen er skitseret i figur 6.10.

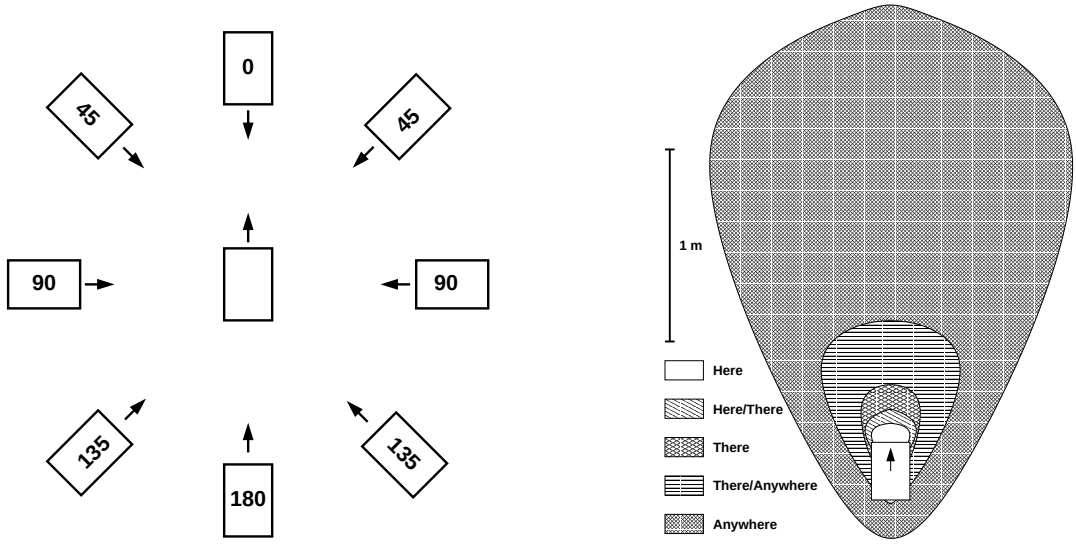


FIGUR 6.9: LEGOs oversigt over de tre zoner *here, there* og *anywhere*

Zone	**Here**		**There**		**Anywhere**	
Vinkel	min	max	min	max	min	max
0°	0	15	7	71	38	212
45°	0	7	3	28	12	91
90°	0	0	0	6	0	24
135°	0	0	0	0	0	9
180°	0	0	0	3	0	23

TABEL 6.1: Oversigt over afstanden mellem grænserne for de tre afstande *here, there* og *anywhere*, målt i forskellige vinkler og afstande for en Spybot.

Tabel 6.1 viser afstanden i centimeter fra den stationære til den mobile Spybot. Den mobile Spybot havde front imod den stationære under hele undersøgelsen. Venstre side af figur 6.10 viser opstillingen for undersøgelsen og højre side er en



FIGUR 6.10: Tv. placering af afsendere i forhold til modtageren i undersøgelse 1. Pilene angiver robottens orientering og tallene er graderne mellem mobil og stationær Spybot. Th. skitsering af zonerne på baggrund af værdierne i tabel 6.1

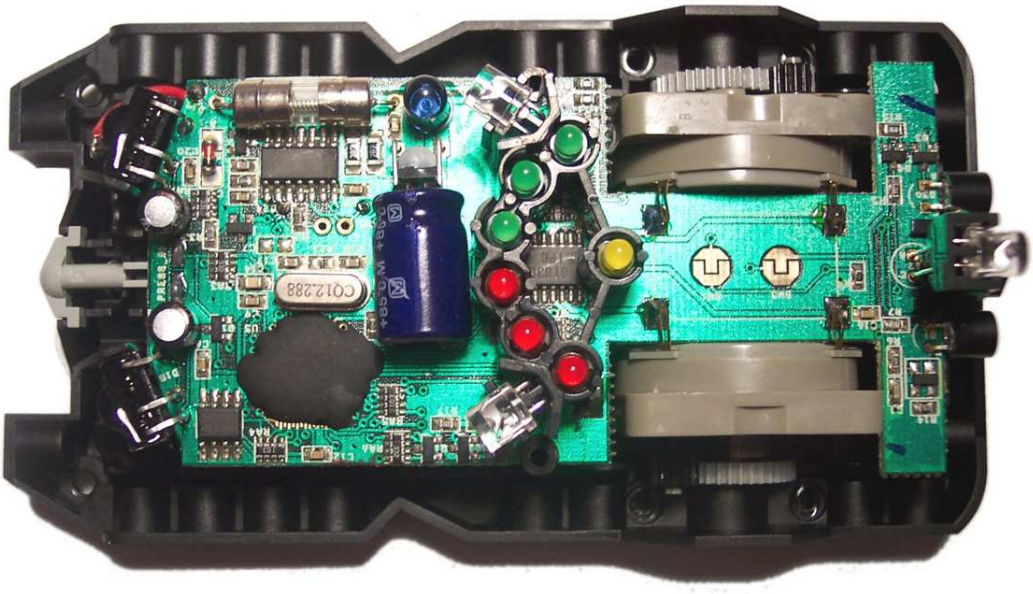
illustration af værdierne fra tabel 6.1. På trods af at undersøgelsen blev udført under gunstige lysforhold, det vil sige intet naturligt lys og ingen lysstofrør, var der stor usikkerhed i afstandsbedømmelsen og derfor er der angivet en minimum og en maksimum værdi for hver zone. Dermed er der et overlap mellem de forskellige zoner, som det også fremgår af figur 6.10.

Det fremgår af Spybot dokumentationen, at en Spybot har to IR modtagere og fire IR sendere. På figur 6.11 ses det, at de to IR modtagere sidder forrest på robotten i højre og venstre side (de to sorte firkanter med tre ben på hver side af den grå trykføler). Derfor giver det god mening, at en Spybot er bedst til at se fremad og skråt til hver side. Der sidder to IR sendere midt på Spybotten pegende skråt til højre og venstre og de to andre sender sidder bag på spybotten, en midt i og den anden sidder lidt skjult ved siden af.

Der er forholdsvist store usikkerheder på afstandsbedømmelsen, men om det skyldes, at pakker går tabt eller om afstandsbedømmelse blot er en usikker affære kan ikke afgøres på baggrund af undersøgelse 1 alene. Derfor vil undersøgelse 2 være en nærmere undersøgelse, af om støj fra omgivelserne påvirker pakker udsendt fra Spybotterne i nævneværdig grad.

6.3.3 Undersøgelse 2

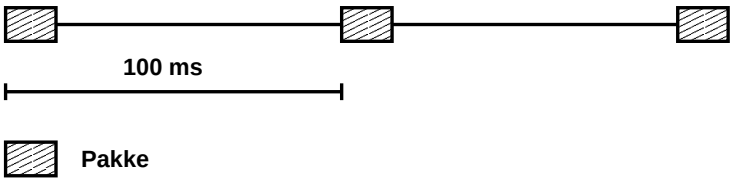
I undersøgelsen er der en Spybot som sender og en der modtager. Sendehyppighed øges for at afdække, hvor mange pakker en Spybot kan modtage uden pakketab. Eksperimentelt har jeg målt, at det maksimale antal pakker en Spybot kan sende



FIGUR 6.11: Spybotten inden i. Venstre side er frem. Billedet er hentet fra [13]

ligger på omkring 10 i sekundet. Det blev gjort ved at lade en Spybot sende 1000 pakker så hurtigt som muligt. Tidstagningen ved dette forsøg blev foretaget manuelt.

Da der kun er en Spybot, der sender noget, er fejlkilden med pakkesammenstød elimineret. Fra afsnit 6.2 ved vi, at det tager 16 ms at sende en datapakke, det betyder at når vi sender med den maksimale hyppighed på 10 pakker i sekundet udnyttes kun 16 % af sendekapaciteten. Et sendeforløb er illustreret i figur 6.12. Tabes der ingen eller kun ganske få pakker betyder det, at IR støj fra omgivelserne kun har en meget lille betydning. Er der et stort pakketab er det umiddelbart vanskeligt at forudsige, om tabet skyldes støj eller at modtageren ikke har kapacitet til at håndtere alle pakker.



FIGUR 6.12: Datapakker, der sendes med en hyppighed på 10 pakker i sekundet.

Opstilling består af to Spybots med omkring 30 centimeter imellem sig. Opstillingen er illustreret i figur 6.13 blot med den forskel, at kun sender 3 og modtageren var en del af undersøgelsen. Resultaterne fra undersøgelse 2 er opsummeret

i tabel 6.2.

pk/sek	Pk sendt	tabt1	tabt2	tabt3
1·1	500	0	0	0
$1 \cdot 3\frac{1}{3}$	1000	0	0	1
1·5	1000	1	0	1
1·10	1000	1	1	0

TABEL 6.2: Oversigt over pakketab ved forskellige sendehyppigheder når en Spybot sender og en Spybot modtager.

Det ses, at pakketabet er højest 0,1 % selv, når der sendes med maksimal hyppighed. Dermed kan støj fra omgivelserne stort set udelukkes som fejlkilde. Desuden ved vi, at en Spybot uden problemer kan håndtere 10 pakker i sekundet. Der er pakketab i to tilfælde, når der sendes 10 pakker i sekundet og der er ikke noget pakketab, når der kun sendes 1 pakke i sekundet, men det kan ikke afgøres om årsagen er målefejl, en svag støjkilde i omgivelserne eller at modtageren i enkelte tilfælde ikke kan følge med. Desuden sendes der flere pakker i de tre nederste målinger, hvorfor fejl lettere kan opstå.

6.3.4 Undersøgelse 3

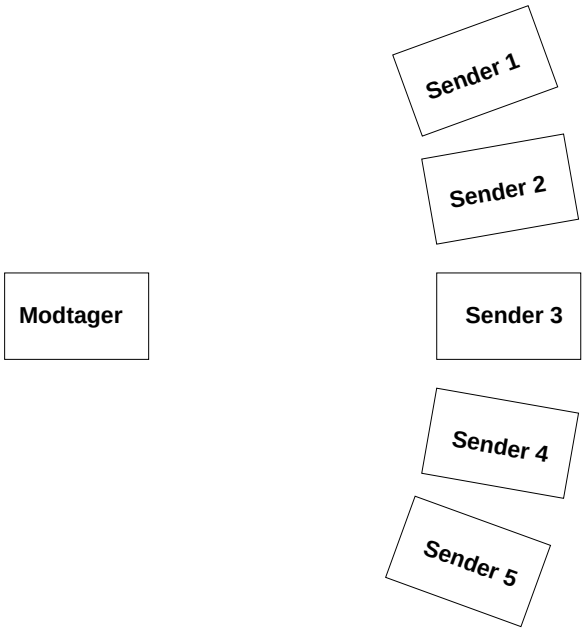
Da fejlkilden fra omgivelsesstøj nu er så godt som elimineret, skal den næste undersøgelse afdække, hvor mange pakker en Spybot kan håndtere, samt om flere Spybots kan sende pakker samtidigt, uden pakkesammenstød forekommer i højere grad. Min hypotese er, at der vil opstå et pakketab, når det sendes tilstrækkeligt hyppigt.

Timerne i legetøjsprodukter er ofte ustabile, det er de bl.a. i RCXen. Usikkerheden gør at timerene måle et tidsinterval som kortere eller længere end det egentligt er og det kaldes at timeren driver. Drivende timere betyder at selv om flere sendere bliver startet, således at deres pakker ikke støder sammen vil deres timere ofte drive en lille smule og dermed kan pakkesammenstød opstå, når to eller flere pakkeforsendelser, der sendes med et fast interval, begynder at sende samtidigt. Min anden hypotese er derfor, at pakketab kan opstå i klumper, når forsendelser fra to eller flere Spybots driver sammen pga. ustabile timere.

I undersøgelsen er der mellem to og fem Spybots, som sender pakker og en, der modtager. Sendehyppigheden øges som i ovenstående undersøgelse. En skematiseret udgave af opstillingen forefindes i figur 6.13. En oversigt af resultaterne af undersøgelse 3 findes i tabel 6.3.

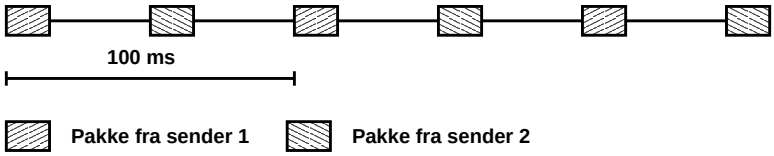
Når flere Spybots sender IR pakker samtidigt, kan der opstå tre forskellige situationer som er:

1. Pakkerne bliver afsendt med en passende afstand uden at støde sammen. Illustreret i figur 6.14

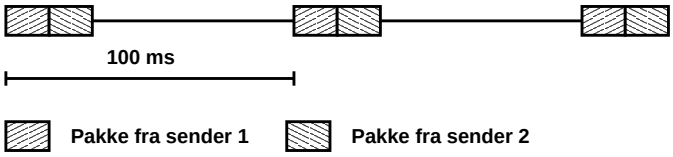


FIGUR 6.13: Illustration af placering af modtager og sendere i undersøgelse 2 og 3

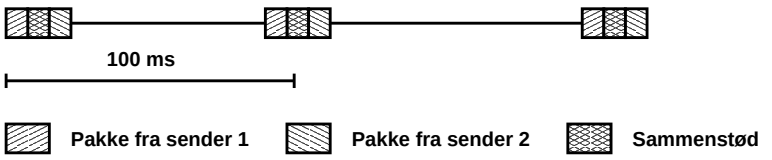
2. Pakkerne kan blive afsendt meget tæt. Illustreret i figur 6.15
3. Pakkerne kan støde sammen og dermed blive ødelagt.Illustreret i figur 6.16.



FIGUR 6.14: Situation, hvor IR pakker sendes med jævnt fordelt afstand. Pakkerne bliver sendt med en hyppighed på 10 pakker i sekundet.



FIGUR 6.15: Situation, hvor IR pakker bliver afsendt meget tæt på hinanden. Pakkerne bliver sendt med en hyppighed på 10 pakker i sekundet.



FIGUR 6.16: Situation hvor IR pakkerne støder sammen. Pakkerne bliver sendt med en hyppighed på 10 pakker i sekundet.

pk/sek	**# Sendere**	**Pk sendt**	**tabt1**	**tabt2**	**tabt3**
2·1	2	1000	44	0	10
$2 \cdot 3\frac{1}{3}$	2	2000	44	63	2
2·5	2	2000	1	6	773
2·10	2	2000	113	2	980
3·1	3	1500	1	0	69
$3 \cdot 3\frac{1}{3}$	3	3000	1	170	657
3·5	3	3000	883	97	940
3·10	3	3000	281	1152	1273
4·1	4	2000	223	1	69
$4 \cdot 3\frac{1}{3}$	4	4000	953	795	442
4·5	4	4000	1890	975	1277
4·10	4	4000	2184	1386	2134
5·1	5	2500	279	908	538
$5 \cdot 3\frac{1}{3}$	5	5000	1579	1640	989
5·5	5	5000	1890	2156	2912
5·10	5	5000	3282	2927	2630

TABEL 6.3: Oversigt over pakketab ved forskellige sendehyppigheder med en modtager og to til fem sendere.

Tabel 6.3 viser måleresultaterne fra undersøgelse 3. Værdien **pk/sek** angiver antallet af afsendte pakker pr. sekund, # **Sendere** angiver antallet af sendere, **pk sendt** er det samlede antal pakker sendt og **tabt1-tabt3** angiver, hvor mange pakker modtageren tabte i måling 1 til 3.

Resultaterne er ganske anderledes end i undersøgelse 2. Allerede ved afsendelse af en pakke i sekundet fra to Spybots opstår der et pakketab, der er markant højere end i undersøgelse 2. Vi ved, at modtageren kan tage imod mindst 10 pakker i sekundet uden nævneværdigt pakketab. Alligevel mister modtageren, helt op til 4,4 % af de afsendte pakker. Det tyder på, at pakkerne støder sammen mellem senderen og modtageren som illustreret i figur 6.16 eller bliver afsendt så tæt, at modtageren ikke kan håndtere alle pakkerne som illustreret i figur 6.15 . Omvendt er der målinger, hvor ingen pakker går tabt, så situationen i figur 6.14 opstår også.

Denne tendens er gennemgående for alle målinger, hvor der er to afsendere og en modtager. Ser man på målingerne i tabel 6.3, hvor to afsendere hver sender 10 pakker i sekundet, er der to målinger, hvor under 0,2 % af pakkerne tabes. Det må være målinger, hvor der ikke er pakkesammenstød under forsendelsen, og hvor der er tilpas afstand mellem pakkerne, som gør, at modtageren kan håndtere alle pakker. Hermed kan det afgøres, at modtageren er i stand til at håndtere mindst 20 pakker i sekundet, hvis modtagerforholdene er de rette.

Længere nede i tabel 6.3 er der nogle målinger, hvor der er blevet sendt 5 pakker i sekundet af 4 sendere, altså en samlet sendehyppighed på 20 pakker sekundet. Her mistes 24-47% af de afsendte pakker. Vi ved, at modtagerkapaciteten er mindst 20 pakker pr. sekund, hvis sendeforholdene er de rette. Da der tabes så mange pakker, er det tegn på, at der enten sker pakkesammenstød eller at pakkerne ankommer med så tæt mellemrum, at modtageren ikke kan nå at håndtere alle pakker. Det er en generel tendens, at når antallet af sendere stiger, så stiger pakketabet. Det kan forklares ved, at jo flere sendere der er, jo større er chancen for pakketab også selv om det samlede antal pakker, der sendes, er mindre en modtagerens modtagerkapacitet.

Ser man på målingerne nederst i tabel 6.3, hvor der sendes fem gange ti pakker i sekundet, skyldes det meget høje pakketab primært sammenstød mellem pakker. I afsnit 6.2 nåede vi frem til, at det tager 16 ms at sende en IR pakke. Når der sendes 50 pakker i sekundet som i den sidste måling, er der dermed IR pakker i luften 80% af tiden. Timeren driver en smule og derfor er det så godt som umuligt at undgå at pakker støder sammen. I denne undersøgelse taber modtageren da også mellem 52% og 65% af de pakker, den burde have modtaget.

Af det store pakketab ses det tydeligt, at der ikke er benyttet en kommunikationsprotokol, der tager hånd om pakketabsproblemer. Det kunne fx være implementeret ved at benytte acknowledgements [26], hvor senderen gensender pakker, hvis modtageren ikke har sendt en kvitering inden for et vist tidsrum.

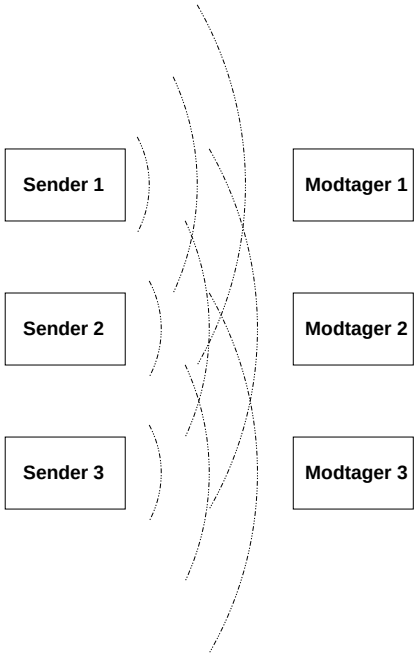
Under undersøgelsen kunne det ofte observeres, at de store pakketab opstod i klumper. Med det menes, at forsendelsen af pakker kunne forløbe uproblematisk i et stykke tid, hvorefter store pakketab pludselig opstod. Efter lidt tid kom forsendelserne til atter at forløbe uden større tab af pakker. Årsagen til det fænomen skyldes formentligt, at senderne som udgangspunkt har sendt pakker med et passende mellemrum som i figur 6.14. Herefter er de på grund af forskelle i sendernes timere kommet til at sende pakker meget tæt eller endda oven i hinanden som vist i figur 6.15 og 6.16.

6.3.5 Undersøgelse 4

Efter at have fået et nogenlunde overblik over, hvor mange pakker en Spybot kan håndtere, samt hvordan mange afsendere af IR pakker påvirker det samlede antal pakker, som modtageren håndterer, er det relevant at undersøge, hvordan situationen ser ud, når der er flere modtagere. Det er relevant, fordi det på baggrund

af ovenstående undersøgelser ikke har været muligt at afgøre, om årsagen til de mistede pakker er pakkesammenstød eller pakker afsendt meget tæt. I undersøgelse 4 er der to Spybots, der sender pakker til hver deres modtager. Det skal bemærkes, at hver modtager er nødt til at håndtere alle pakker for at finde ud af, om den selv er modtageren. Som i undersøgelse 2 og 3 øges sendehyppigheden for at undersøge, om andre robotters pakker forstyrrer modtagelsen af pakker i højere grad, når der sendes flere pakker.

Forsøgsopstillingen ligner den der er illustreret i figur 6.17 blot med den ændring, at det kun er sender og modtager 1 og 2, der er med i forsøget.



FIGUR 6.17: Illustration af placering af modtager og sendere i undersøgelse 4 og 5

pk/sek	**pk/sender**	**Modtager 1**			**Modtager 2**		
		M1	M2	M3	M1	M2	M3
1	500	1	34	0	0	365	274
$3\frac{1}{3}$	500	0	0	0	0	0	0
5	1000	2	0	0	0	3	1
10	1000	349	420	1	0	2	1

TABEL 6.4: Oversigt over pakketab ved forskellige sendehyppigheder, hvor to Spybots sender pakker til to modtagere.

Måledata for undersøgelse 4 er samlet i tabel 6.4. Værdien **pk/sek** er som ovenfor; **pk/sender** angiver, hvor mange pakker hver sender har sendt, **Mod-**

tager 1 og **Modtager 2** angiver, hvilken af de to modtagere der er tale om og **M1-M3** angiver, om der er tale om måling 1, 2 eller 3. Tallene under M1-M3 angiver, hvor mange pakker en modtager har mistet i den pågældende måling på samme måde som tabt1-tab3 ovenfor.

Ser man på resultaterne, når hver sender Spybot afsender 1 pakke i sekundet, viser måling M1 at begge modtagere modtager stort set alle deres pakker. M3 viser at modtager 1 får alle sine pakker, men modtager 2 mister over halvdelen af de pakker, som den burde have modtaget. Det kan forklares, ved at pakkerne er blevet afsendt meget tæt på hinanden, som vist i figur 6.15. Pakkerne til modtager 1 er blevet afsendt først og dem til modtager 2 lige efter. Modtager 2 er nødt til at undersøge den først afsendte pakke for at afgøre, hvem pakken er til. Når næste pakke ankommer, bliver den ikke håndteret. Det kan fx skyldes manglende bufferplads.

Årsagen til at det kan udelukkes, at pakkerne er stødt sammen i M3, er, at ved pakkesammenstød bliver begge pakker ødelagt og begge modtagere ville dermed have mistet pakker.

Måling M2 ved en sendehyppighed på 1 pakke i sekundet, kunne derimod godt tyde på en kombination af sammenstød og pakker afsendt tæt efter hinanden, da begge modtagere mister pakker.

Undersøgelse 4 kaster lys over, hvordan pakker afsendt samtidigt forstyrrer hinanden. I den femte og sidste undersøgelse vil dette aspekt blive uddybet yderligere.

6.3.6 Undersøgelse 5

Undersøgelse 5 er som undersøgelse 4 med den forskel, at her benyttes 3 sendere og 3 modtagere. Måledata fra undersøgelse 5 er samlet i tabel 6.5.

pk/sek	**pk/sender**	**Modtager 1**			**Modtager 2**			**Modtager 3**		
		M1	M2	M3	M1	M2	M3	M1	M2	M3
1	500	2	126	258	0	0	202	0	330	0
$3\frac{1}{3}$	500	67	0	0	0	373	0	420	86	1
5	1000	3	405	15	0	37	428	2	305	0
10	1000	1	9	511	43	756	6	742	2	446

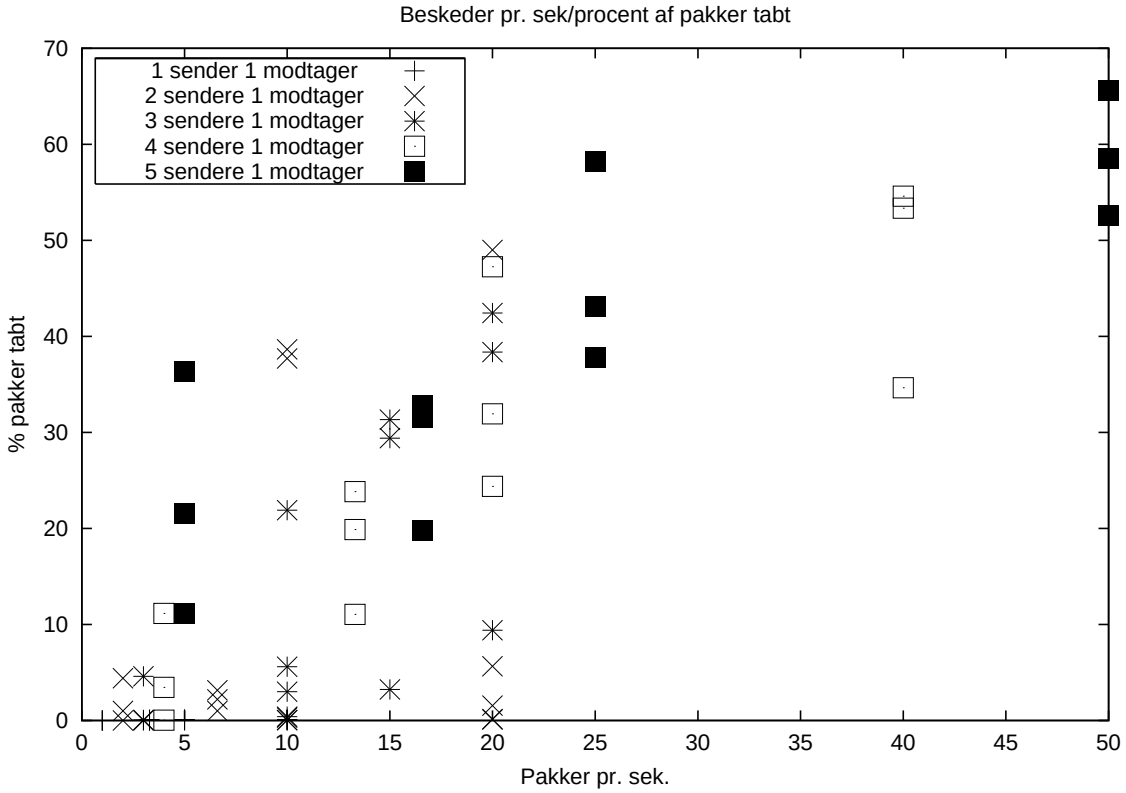
TABEL 6.5: Oversigt over pakketab ved forskellige sendehyppigheder hvor tre Spybots sender pakker til tre modtagere.

I undersøgelse 5 bliver resultaterne fra undersøgelse 3 bekræftet: jo flere sendere der er og jo højere hyppighed de sender med, jo færre pakker når frem. Dog bliver det tydeligere, hvorfor beskederne forsvinder. Ser man på de nederste målinger i 6.5, hvor hver afsender sender med en hyppighed på 10 pakker i sekundet, ser man for måling M2, at modtager 1 og 3 modtager stort set alle de pakker

de bør, hvorimod modtager 2 mister 75% af alle sine pakker. I M1 er tendensen tilsvarende, modtager 1 og 2 modtager de fleste pakker, de bør, men modtager 3 mister omkring 74%. M3 tyder på, et massivt sammenstød af pakker, da alle modtagere her mistede 45% eller flere af de pakker, de burde have modtaget.

6.4 Konklusion på undersøgelser

Undersøgelserne har kastet lys over, hvorfor det til tider kan knibe for Spybots at finde hinanden og reagere på fjernbetjeningstryk. Det skal endnu en gang nævnes at undersøgelserne er gennemført med datapakker, der har en størrelse på 7 bytes, hvor pingpakkernes størrelse er 4 bytes. Dog er det overvejende sandsynligt at tendensen for datapakker og pingpakker er den samme, når det kommer til sammenhængen mellem sendehyppighed, antallet af sendere og pakketab.



FIGUR 6.18: Grafisk oversigt over måleresultaterne fra tabel 6.2 6.3, der viser hvor stor procentdel af det samlede antal pakker, der forsvinder, når sendehyppigheden øges.

I undersøgelserne 2 til 5 er den generelle tendens at jo flere pakker der bliver sendt til en Spybot, jo større procentdel af pakker bliver der tabt. En anden tendens er, at jo flere Spybots, der sender samtidigt, jo flere pakker går der tabt

for hver enkelt modtager. Årsagen til at pakketabet stiger med pakkemængden skyldes formentligt, dels at modtagerens øvre grænse for hvor mange pakker der kan håndteres, bliver overskredet, dels at antallet af pakker, der sendes så tæt at modtageren mister nogle, stiger. Samtidigt øges risikoen for pakkesammenstød, når antallet af afsendere og antallet af afsendte pakker øges.

Dermed ved vi også, at LEGO ikke umiddelbart har implementeret en strategi til håndtering af pakketab. Det kunne være gjort ved at implementere en simpel ALOHA protokol [57] s. 471. Tendensen ses tydeligt i figur 6.18, hvor målingerne fra tabel 6.2 og 6.2 er indtegnet som fem typer punkter. Hver gruppe af punkter angiver, hvor mange sendere der er pr. modtager.

Når der ikke er implementeret en protokol, der håndterer pakketab er årsagen sandsynligvis at en sådan protokol vil øge mængden af pakker, der bliver sendt frem og tilbage, da modtageren af pakkerne vil være nødt sende en kvittering tilbage til senderen for modtagelsen af pakken. På den måde vil protokollen være med til at bidrage til det det problem, den selv forsøger at afhjælpe; nemlig at forhindre pakketab.

Har vi fx et scenarie, hvor fem Spybots kører rundt mellem hinanden og skal finde hinanden, så ved vi, at Spybots som udgangspunkt udsender 4 pingpakker i sekundet, der har en størrelse på 44 bits og dermed tager lidt over 9 ms at sende. Dermed udnyttes 18 % af den mulige sendekapacitet. Sammenlignes det med resultatet fra tabel 6.3, hvor 5 Spybots hver sender 1 pakke i sekundet og dermed kun udnytter 8 % af mediets kapacitet, oplever vi alligevel et pakketab på mellem 11% og 36%. Holder vi det sammen med resultaterne fra undersøgelse 4 og 5 er det ligeledes klart, at nogle Spybots i dette spilscenarie vil modtage næsten alle pakker, hvorimod andre risikerer at miste størstedelen af deres pakker.

På baggrund af undersøgelsen af Spybotternes sende- og modtager forhold er det klart, at hvis Spybots skal benyttes i et robotspil, bør der i dette spil kompenseres for Spybotternes svagheder, eller eventuelt kan de udnyttes til spillets fordel. Resten af specialet vil handle om at identificere, implementere og afprøve et sådant spil.

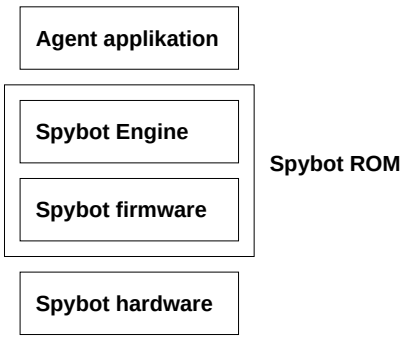
7 Implementering af robotspil

I dette kapitel beskrives en oversigt over Spybottens arkitektur. Dernæst vil arkitekturen for kontrolprogrammet til de autonome adaptive agenter blive gennemgået og designovervejelserne, der danner baggrunden for valget af arkitektur, vil blive motiveret.

7.1 Gennemgang af LEGO Spybottens arkitektur

Spybottens in- og output muligheder blev nævnt i afsnit 5.3.5 og forhold omkring sende- og modtagerforhold for Spybots blev gennemgået i kapitel 6.

Figur 7.1 viser lagene for Spybottens software arkitektur. I dette afsnit vil de to midterste lag kaldet **firmware** og **Spybot Engine**, der begge ligger i Spybottens ROM, blive gennemgået.



FIGUR 7.1: Oversigt over lagene i LEGO Spybotics Arkitekturen.

Begge lag er beskrevet i [61]. Hardwarelaget er allerede blevet behandlet i det omfang, det er nødvendigt og applikationen, der er det øverste lag på stakken, vil blive gennemgået i afsnit 7.2.

7.1.1 Firmware lag

Figur 7.2 er en oversigt over den firmware, der ligger i Spybottens ROM. Der er enkelte elementer, der skal fremhæves. I øverste venstre hjørne i blokken **Storage** er der en **EEPROM**[1], der er et persistent lager, hvor variabler kan gemmes mellem batteriskift.

World relation table i blokken **Functional blocks** er også central i forbindelse med skabelsen af autonome agenter baseret på LEGO Spybots. World

[1]"An EEPROM, or Electrically-Erasable Programmable Read-Only Memory, is a nonvolatile storage chip used in computers and other devices[...] It may be erased and reprogrammed only a certain number of times, ranging from 100,000 to 1,000,000, but it can be read an unlimited number of times." Forklaringen stammer fra [2]

relation tabellen er en tabel med 16 indgange, hvor der er information om andre Spybots i nærheden af denne Spybot. Informationen består blandt andet af de andre Spybotters ID samt afstand og retning til dem. Et felt i hver indgang bliver benyttet til opbevaring af den brugerdefinerede information, som bliver sendt med pingpakkerne (jf. afsnit 6.1.1)[2]. Desuden er der et 4 bits felt i hver indgang, der kan benyttes til noter. World relation tabellen opdateres hver gang, en pingpakke modtages.

Nederst er en række **Events**, der kan benyttes i forbindelse med programmering af en agent, blandt andet et **bump event**, der bliver udløst ved tryk på tryksensoren.

7.1.2 Spybot Engine lag

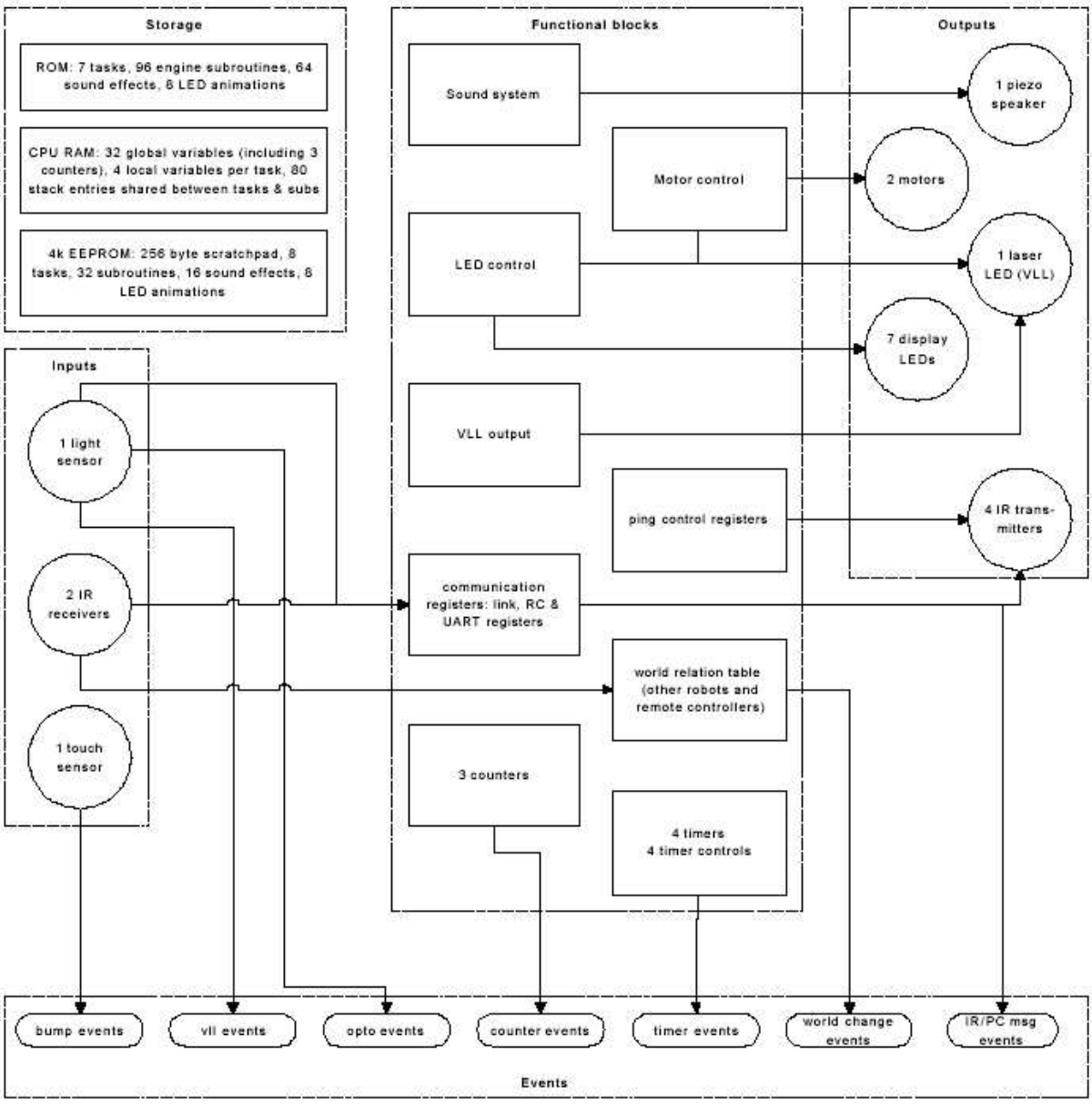
Benytter man LEGO's *Spybot Engine* giver man afkald på en del variabler som *Spybot Engine* benytter som registre og løbende holder opdateret. Der er dog ingen grund til at lade *Spybot Engine* opdatere variabler, som man ikke har brug for, så de variabler kan slettes og benyttes til andet formål. *Spybot Engine* overvåger eventregistrene og sørger for at passende brugerdefinerede subrutiner bliver kaldt i tilfælde af events. Bliver trykføleren påvirket kaldes subrutinen *sBumpAction* og tilsvarende for de andre events. Af variabler som *Spybot Engine* holder opdateret, er blandt andet en, der holder styr, på hvilken Spybot der er *target* og en anden indeholder information om, hvilken zone dette *target* befinder sig i.

Jeg besluttede, at agentprogrammet til Spybots skulle skrives i *MindScript*, som er et programmeringssprog udviklet af LEGO. Ved at programmere i *MindScript* er det muligt at benytte de mange subrutiner i ROM'en, som LEGO har skrevet til LEGO Spybots. Desuden er det muligt at benytte LEGO *Spybot Engine*, som stiller mange nyttige funktionaliteter til rådighed blandt andet i forbindelse med at bedømme afstand og retning til andre Spybots. Bedømmelse af afstand og retning til andre Spybots er en forudsætning for at kunne implementere en nogenlunde fornuftig autonom agent. Figur 7.3 viser et udsnit af et agentprogram skrevet i *MindScript*, i LEGO's editor *ScriptEd*.

Andre muligheder for valg af programmeringssprog er *Not Quite C*, [14] der ikke på nuværende tidspunkt understøtter Spybotternes afstands og retningsbestemmelser. Derfor er det blevet fravalgt til at skrive kontrolprogrammet til modstanderrobotterne i.

Spybotternes grafiske programmeringsmiljø, er for simpelt til at kunne implementere en evolutionær algoritme. Her er der kun mulighed for at vælge mellem et begrænset antal foruddefinerede **adfærdskapsler**, der kan placeres på skærmbilledet og derved bestemmes det, hvordan en Spybot skal reagere på andre Spybots og på tryk på fjernbetjningen. Figur 7.4 viser et screenshot fra den del af den grafiske Spybotics applikation, hvor det er muligt at vælge adfærd for en Spybot ved

[2]Det felt benytter jeg til at angive Spybottens hold

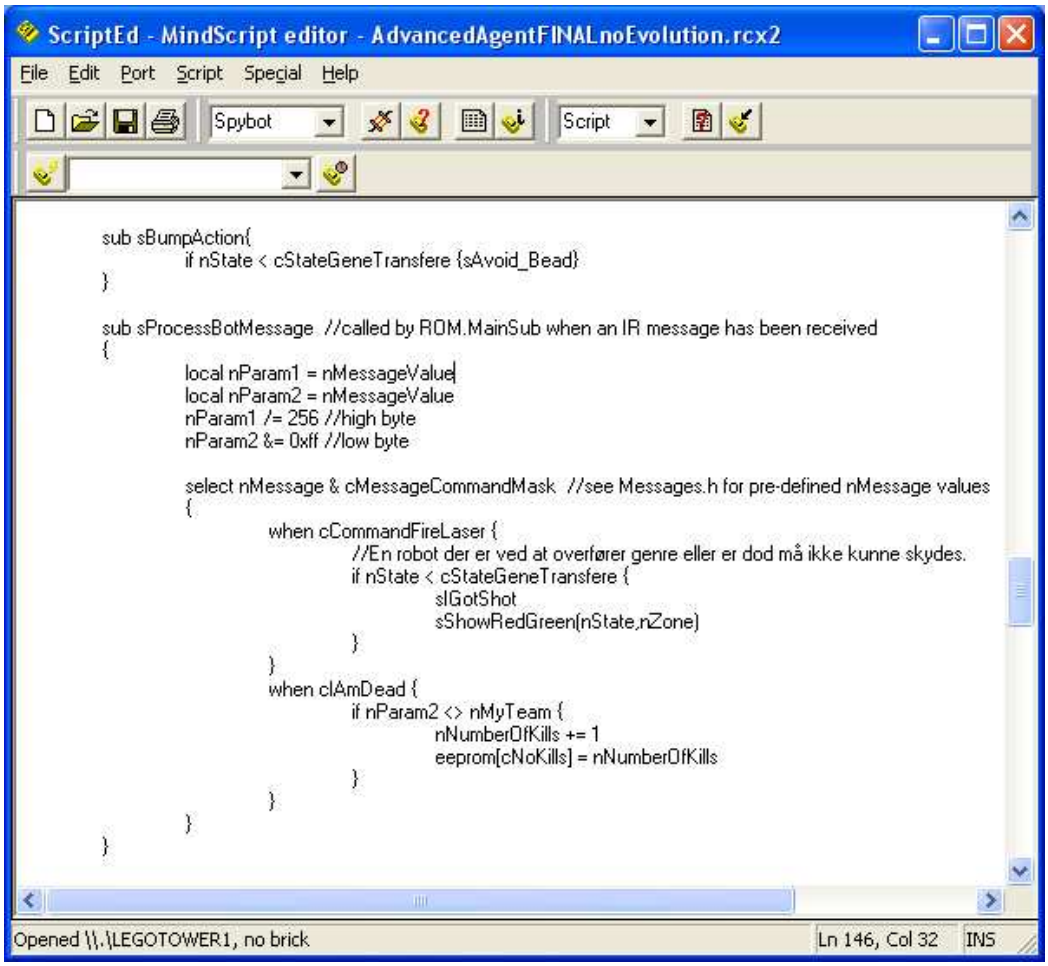


FIGUR 7.2: Oversigt over firmwarelaget i LEGO Spybotics. Figuren stammer fra [61]

mødet med andre Spybots. Figur 7.5 viser et skærmbillede hvor det bestemmes, hvordan en Spybot skal reagere på fjernbetjeningstryk.

Til sidst er der assemblysproget *LASM*, som det vil tage meget længere tid at implementere programmet i. Figur 7.6 viser et *LASM* program, skrevet i LEGOs editorprogram *ScriptEd*.

Ønsker man at benytte LEGOs *Spybot Engine* må man som programmør finde sig i at dele af ens kode er mørklagt. Det skyldes, at koden til LEGOs *Spybot En-*



FIGUR 7.3: Screenshot af *MindScript* programme fra editoren *ScriptEd*.

gine er en forretningshemmelighed og dermed utilgængelig for andre end LEGOs ansatte. Det betyder, at der kan forekomme uforudsigelig adfærd i kontrolprogrammer, der bygger på LEGOs *Spybot Engine*. Jeg har skrevet programmer til Spybots både med og uden brug af *Spybot Engine* og nåede frem til, at fordelene ved at benytte den, for mit vedkommende, overstiger ulemperne.

7.2 De autonome agenters kontrolprogram

I dette afsnit gennemgås arkitekturen af de autonome Spybotters kontrolprogram. I valget af arkitektur er der hentet inspiration fra Thiemo Krinks artikel *Motivation Networks - A Biological Model for Autonomous Agent Control* [39].

Motivationsnetværk er en biologisk inspireret måde at implementere adaptive autonome agenter på [39] [44]. Dyr er motiveret til forskellige handlinger på forskellige tidspunkter. En spurv kan således være motiveret for handlinger

FIGUR 7.4: Screenshots fra den grafiske Spybotics applikation hvor der kan vælget adfærdskapsler der bestemmer hvordan en Spybot skal reagere overfor andre Spybots.

som spise, sove, flygte, yngle med videre. Motivationen for forskellige handlinger ændres afhængig af spurvens omgivelser og tilstand; fx er spurven meget lidt motiveret til at sove når den lige er vågnet. Motivationen for at spise stiger jo længere siden, det er, at den har spist sidst og dermed jo lavere, dens energidepoter er. Er der et rovdyr i nærheden overtrumfer motivationen for at flygte andre motivationer. Et dyr handler hele tiden efter det, den højeste motivation dikterer. Pointen med motivationsnetværk er at give computerbaserede agenter en hensigtsmæssig eller måske endda intelligent adfærd. I Krinks model har hvert individ en række gener, der hver især oversætter fra ydre og indre stimuli til motivation. Fx vil lavt energiniveau, hvilket er det samme som stor sult, forekomsten af mad og fraværet af rovdyr, formentlig resultere i høj motivation for at indtage føde. Ideen med gener er, at når et individ formerer sig, så vil individets gener overføres til næste generation. Individer med dårlige gener; fx stor motivation for at sove, når der er rovdyr til stede vil uddø og til slut vil der være en population af stærke individer tilbage.

Tilgangen virker på forhånd ideel til autonome robotmodstandere i et robotspil. Modstanderen skal have motivationer som angrebslyst, lyst til at flygte og

FIGUR 7.5: Screenshots fra den grafiske Spybotics applikation, hvor der kan vælges *adfærdskapsler*, der bestemmer hvordan, en Spybot skal reagere på fjernbetjningstryk.

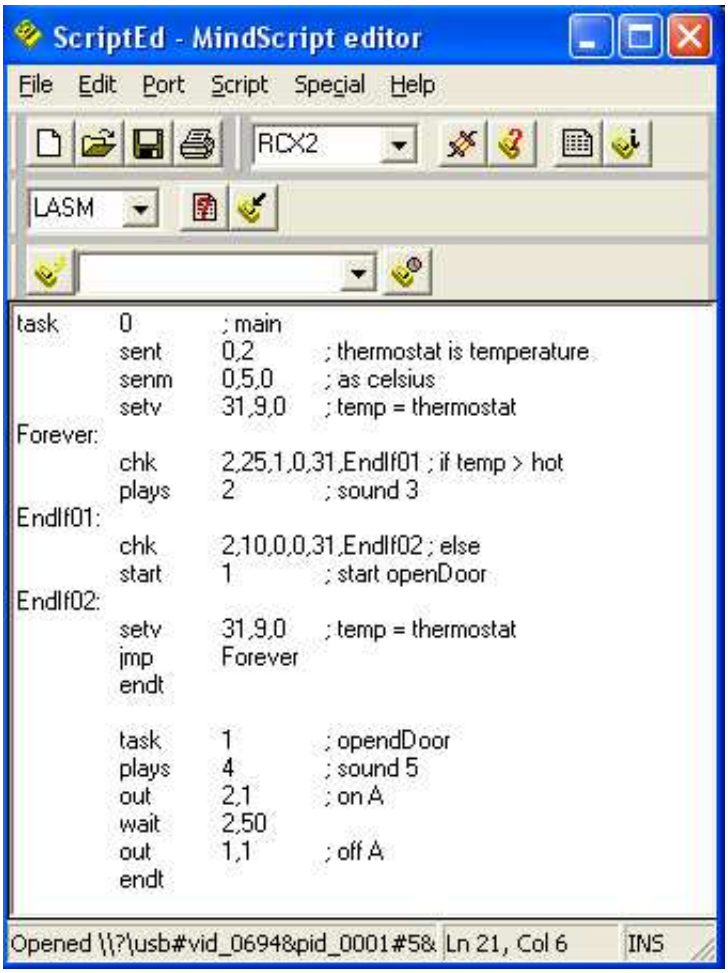
lyst til at afsøge.

I *Evolution of Computer Bugs - an Interdisciplinary Team Work* [21] gjorde Caprani og Fredslund m.fl. en indsats for at give adfærdsbaserede robotter en interessant adfærd. Deres arbejde er der ligeledes blevet skelet til forbindelse med konstruktion af kontrolprogrammet.

I kapitel 4 blev forskellige tilgange til implementering af adaptive agenter gennemgået. *Embodied Evolution* [23, 63] blev senere i kapitlet nævnt som et eksempel på, hvordan evolutionære robotter kan fungere i praksis. Dele af fremgangsmåden virker umiddelbart fornuftig i forbindelse med adaptive robotmodstandere.

På baggrund af dette samt overvejelserne omkring agenter generelt og i spil fra afsnit 4.1, kom arkitekturen for kontrolprogrammet for en autonom modstander Spybot til at se ud som vist i figur 7.7.

Figur 7.7 består af en række blokke. Først vil den evolutionære algoritme der danner grundlag for det adaptive element blive gennemgået. Herefter vil blokkenes funktion og sammenhængen blive gennemgået, og til sidst vil enkelte komponenter i blokkene blive gennemgået i mere detaljeret.

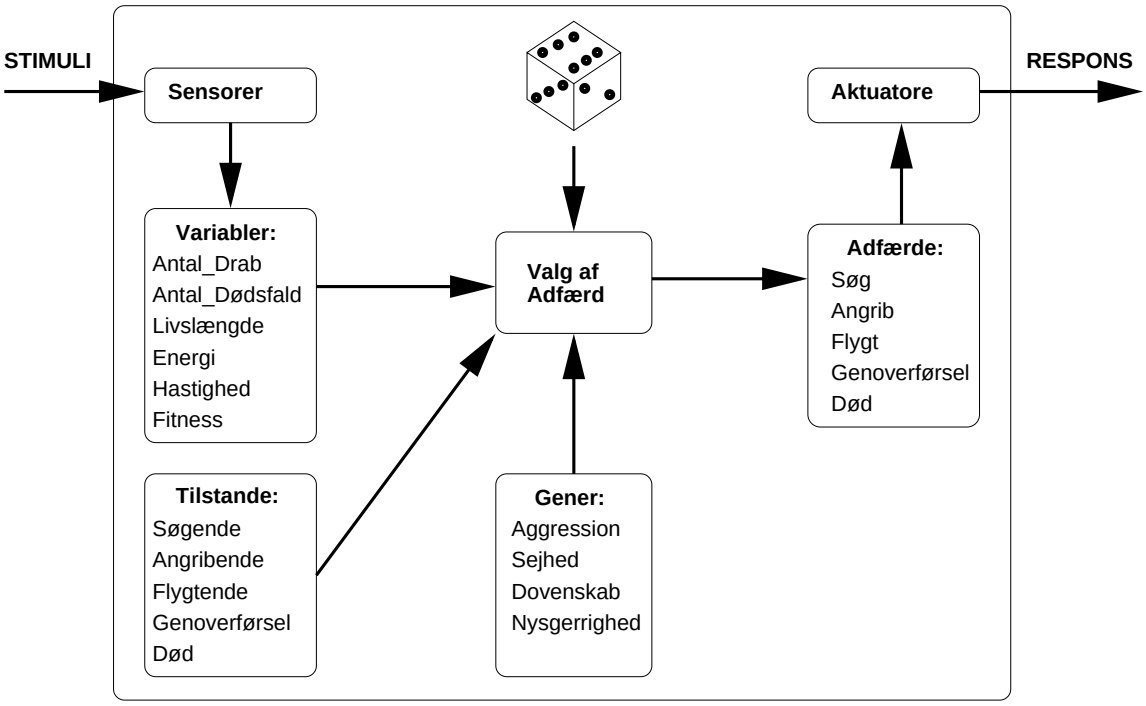


FIGUR 7.6: Screenshot af *LASM* program, skrevet i editoren *ScriptEd*.

7.2.1 Den evolutionære algoritme

I afsnit 4.2.3 blev det nævnt, at de grundlæggnede egenskaber for evolutionære algoritmer er flere individer, flere generationer, krydsning mellem individer, mutation og en fitnessværdi, som det enkelte individs succes kan afgøres på baggrund af.

Den evolutionære algoritmes opgave er at finde den bedste kombination af de fire gener *Aggression, Sejhed, Dovenskab* og *Nysgerrighed*, som er vist i figur 7.7. Generne er hver et tal mellem 1 og 100, hvilket giver i alt 10^8 forskellige kombinationer. Der er altså tale om relativt mange forskellige løsninger, men det er nærliggende, at antage at effekten af en del af dem er stort set identisk. Ændres aggressionsgenet eksempeltvist en smule betyder det at den er en lille smule mere aggressiv eller tilbageholden. Det er næppe sandsynligt at en sådan ændring vil påvirke den ydre adfærd synderligt.



FIGUR 7.7: Arkitekturen for de autonome Spybotters kontrolprogram

For at finde og evaluere gener, eller løsninger om man vil, fra hele søgelandskabet, benytter evolutionære algoritmer sig af de to forskellige redskaber; mutation og krydsning af individer [38]. Mutation benyttes til at tilføje mindre ændringer til generne, så der foretages en lokal søgning. Derfor er det vigtigt, at et eller flere gener muteres i forbindelse med genfødsel af individer, der ikke har klaret sig tilstrækkeligt godt. I algortmen er chancen for mutation sat til 25% pr. gen, hvilket betyder, at et eller flere gener muteres i de fleste tilfælde, hvor mutationsfunktionen kaldes. I algoritmen implementeret i de autonome Spybots benyttes en mindre mutation, der er fordelt omtrent som en Gaussklokke, hvilket er normen inden for evolutionære algoritmer[3].

Krydsning af to individer benyttes for at finde nye løsninger andre steder i søgelandskabet; altså en global søgning. Som krydsning benyttes den naturlige krydsning, der kan opstå, når en Spybot sender sine gener til en anden. Spybotten sender sine gener i tre forskellige pakker. Da pakketabet hos Spybots er relativt stort, når der er mange Spybots til stede, er der vis sandsynlighed for, at en eller flere pakker tabes, når generne udveksles og dermed får modtageren blandet sine egen gener med senderens gener. Til tider bliver indvider genfødt med helt nye gener, hvilket understøtter global søgning.

[3]Tilnærmelse til en Gaussfunktion opnås ved at generere en række tilfældige tal inden for det ønskede interval, summere de tilfældige tal og dele med antallet af genererede tal.

Fitnessværdien skal afspejle i hvor høj grad, et individ opfylder sin opgave. Fitnessværdien for de autonome Spybots beregnes som vist i nedenstående ligning.

$$fitness = 50 + 10 \cdot (Antal\_Drab - Antal\_Dødsfald) + min(Livslængde, 9)$$

Overvejelserne for beregning af fitness var, at lang levetid er posetivt og skal derfor forhøje fitnessværdien en smule. Antal dræbte modstandere og antallet af dødsfald skal veje tungt og derfor henholdsvist forhøjes eller mindskes fitnessværdien i højere grad, når disse værdier ændres. Den ovenstående fitnessværdi giver dog intet bud på, om Spybotten er en underholdende modspiller i et robotspil, men bør give en indikation af, om den tilfredsstillende opfylder sin opgave med at nedkæmpe sine modstandere og selv overleve.

Flere individer er ligeledes centralt i en evolutionær algoritme, da de gode gener opstår som samspil mellem flere individer. Evolutionære algoritmer, der afvikles på PC'er, benytter ofte 100 eller flere individer. Så mange Spybots har jeg ikke til rådighed og selv om det var tilfældet ville der opstå sammenstød af pakker i en grad, så de alligevel ikke ville kunne kommunikere fornuftigt [46]. Antallet af individer i de gennemførte afprøvninger vil være 4-10.

Generationer benyttes heller ikke på samme måde som i en evolutionær algoritme, der bliver afviklet på en PC. Her afsluttes en generation, når hvert individs gener er blevet evalueret ved at indsætte dem i fitnessfunktionen.

For Spybots der bevæger sig rundt i den fysiske verden og forsøger at løse en opgave, er det ikke muligt på et hvilket som helst tidspunkt at afgøre, hvilke individer der klarer sig godt ved at sætte bestemte værdier ind i fitnessfunktionen. En Spybot skal have fungeret i sit miljø et vist stykke tid, inden den har indsamlet erfaringer nok til, at det giver mening at udregne en fitnessværdi. Derfor har hvert individ mulighed for at forny deres gener, når de er blevet nedkæmpet af en modstander, for her er det naturligt at antage at individet ikke opfylder sin opgave godt nok, for ellers var det ikke dødt. Når en Spybot er død idømmes den 30 sekunders karantæne og mens den er ude af spillet, sender den besked om, at den har brug for nye gener, da de gener, den har nu, ikke var nok til at sikre dens overlevelse. Den sendte besked indeholder også Spybottens fitness og hvis en anden Spybot med højere fitness kommer forbi, sender den sine gener til den døde Spybot. Herefter udskifter Spybotten sine egne gener samt fitness ud med de nye gener og fortsætter med at anmode om nye gener indtil karantænetiden er udløbet. Får den ingen nye gener fra andre, genfødes den med sine egne gener med mindre dens fitness er under lavmålet (30) så får den et nyt sæt gener. Inden genfødsel muteres generne.

7.2.2 Sammenhæmg mellem blokkene i figur 7.7

Som det ses på figur 7.7 modtager Spybotten stimuli fra sine omgivelser. Disse stimuli kan være rå input (talværdier) fra trykføler eller lyssensor, eller beskeder fra andre aktører der modtages via Spybottens infrarøde modtagere.

Spybottens variabler opdateres på baggrund af input, og dens adfærd kan ændres på baggrund af variablernes opdatering samt dens nuværende tilstand og gener.

Hvis en Spybot modtager besked om, at den er blevet skudt af en modstander, så ændres dens energiniveau. Falder energiniveauet til nul eller mindre, ændres adfærden eller mangel på samme til *død*. Sker der blot en formindskelse af energiniveauet, bliver det på baggrund af Spybottens genetiske anlæg afgjort, om den skal flygte eller angribe. Støder en Spybot ind i en forhindring, undviger den forhindringen, med mindre den er i tilstanden *død* eller er ved at overføre gener.

Når en adfærd er blevet valgt bliver den omsat til ydre handling via aktuatorerne. Er det eksempelvist blevet afgjort, at Spybotten skal flygte, vil motorerne blive sat til at køre i modsat retning af dens *target*. Hvis Spybotten derimod skal angribe, kører den imod sin modstander og skyder, når den er indenfor *here* zonen.

7.2.3 Sensorer

Spybottens tryk- og lysføler samt dens IR modtager.

7.2.4 Variabler

Spybotterne har en række variabler, der bliver benyttet til at holde styr på forfatningen af Spybotten og dens omgivelser.

Energi Angiver Spybottens energiniveau. Hvis variablen falder til nul eller derunder, dør den.

Hastighed Spybottens hastighed. Der er tre forskellige hastigheder: hurtig, normal og langsom.

Antal_Drab Antallet af dræbte modstandere. Denne værdi indgår i beregningen af fitnessværdien.

Antal_Dødsfald Antallet af gange Spybotten selv er blevet dræbt. Værdien indgår ligeledes i beregningen af fitness.

Livslængde Angiver Spybottens livslængde og er også en komponent i fitnessværdien.

Fitness Fitness variablen angiver Spybottens fitnessværdi. Fitnessværdien for den evolutionær algoritme er angivet i afsnit 7.2.1.

Når en Spybot genfødes med nye gener, nulstilles variablerne *levetid*, *antal_ dræbte* og *antal_ dødsfald*, da disse variabler knytter sig til et individ med et bestemt sæt gener. De resterende variabler initialiseres ligeledes ved genfødsel[4].

7.2.5 Tilstande

Spybottens tilstand afspejler, hvilken adfærd den er i færd med at udføre. De har tilstandene *Søgende*, *Angribende*, *Flygtende*, *Genoverførsel* og *Død*. De enkelte adfærde gennemgår i afsnit 7.2.7 nedenfor.

Spybotten reagerer forskelligt på input alt efter, hvilken tilstand den er i. Er den i søgende tilstand, så undviger den, hvis trykføleren påvirkes. Er den derimod død eller i gang med at overfører gene, så reagerer den ikke på påvirkning af trykføleren.

7.2.6 Gener

Spybotten har fire gener, der har indflydelse på Spybottens valg af adfærd. Gener er heltal mellem 0 og 100 og bliver betegnet som følger:

Aggression angiver viljen til angreb. Dette gen bestemmer om et individ ofte angriber eller trækker sig tilbage. Høj aggression betyder stor sandsynlighed for at angribe, hvorimod lav værdi betyder høj sandsynlighed for at flygte.

Sejhed angiver Spybottens viljen til at angribe, når den er såret. Med såret menes der om variablen *energi* er mindre end det maksimalt mulige. Genet afgøre altså, om et individ skal være tilbageholdende, hvis det er såret. En høj værdi betyder, at Spybotten ofte angriber selv om den er såret, hvorimod en der har lavt genetisk anlæg for at være sej ofte vil trække sig tilbage.

Dovenskab angiver hastigheden et individ bevæger sig med. Høj værdi betyder en meget doven Spybot og jo mere doven den er, jo langsommere bevæger den sig.

Nysgerrighed. Meget nysgerrige Spybots bevæger sig i zig-zag bevægelser og drejer meget rundt omkring sig selv. Er en Spybot meget lidt nysgerrig kører den mest lige ud.

Generne for aggression og sejhed knytter sig til Spybottens adfærd i tilfælde af, at den har set en modstander. Genet for nysgerrighed har betydning, når Spybotten søger modstandere. Dovenskabsgenet har betydning både ved mødet med modstandere og når Spybotten søger.

[4]Det blev dog ændret i kampene mellem to grupper af Spybotter senere

7.2.7 Valg af adfærd

På baggrund af et individs tilstand, gener og værdien af dets variabler afgøres sandsynligheden, for at Spybotten vil udvise en bestemt adfærd. Har en autonom Spybot eksempelvis set en modsander har den to muligheder for at handle; den kan angribe eller den kan trække sig tilbage. Har den genetisk anlæg for at være meget aggressiv, men har lille genetisk anlæg for sejhed, så vil førstnævnte gen favorisere et angreb. Genet for sejhed vil kun spille ind, hvis den er såret. En såret Spybot, der ser en modstander vil være tilbøjelig til at trække sig tilbage. På baggrund af gener og tilstande beregnes sandsynligheden for henholdsvis et angreb og en tilbagetrækning.

Når det bestemmes, om en Spybot skal trække sig tilbage eller angribe, når en modstander kommer inden for rækkevidde, benyttes ligningen nedenfor:

$$Angrebs\_chance = \frac{Aggression + Max(Sejhed, Energi)}{2}$$

Ligningen udtrykker Spybottens chance for at angribe i tal fra 0 til 100. *Aggression* og *Sejhed* er begge gener. En agressiv Spybot som samtidigt er sej eller har sin *Energi* i behold vil næsten altid angribe. Er en Spybot såret, men er sej, vil den stadig være angrebslysten, men er den knap så sej, falder angrebslysten når, dens *Energi* aftager.

Hver gang der sker en ændring af den indre tilstand, fx at Spybotten bliver såret, beregnes der ny *angrebs_ chance* og der vælges atter adfærd.

7.2.8 Adfærd

De autonome har en række adfærde, der vælges på baggrund af deres gener, deres nuværende tilstand deres variabler samt et tilfældigt element. Adfærdene er handlingsmønstre, der udføres under forskellige omstændigheder.

Søg: Søgende adfærd, hvor Spybotten leder efter modstandere. Ser den sit mål inden for zonerne *here* eller *there* vil Spybotten skifte adfærd til enten *flygt* eller *angrib*, alt efter hvor sej og agressiv Spybotten er, samt om den er såret eller ej.

Angrib: Spybotten kører hen til imod sit mål og skyder. Den forfølger, hvis dens modstander forsøger at flygte. Lykkes det modstanderen at slippe ud af dens synsfelt, skifter adfærden til søgende.

Flygt: Spybotten trækker sig tilbage, indtil dens modstander er i zonen *anywhere*, hvor dens modstander ikke kan ramme den med sine skud.

Død: Når en Spybot er blevet skudt tilstrækkelige mange gange (3) uden at have haft chance for at restiturere sig, så dør den. En Spybot, der ikke er blevet ramt af skud i 30 sekunder restiturer $\frac{1}{3}$ af sin Energi. At en Spybot dør vil

sige, at den ikke deltager i spillet i en given tidsperiode (30 sekunder). Mens den er ude af spillet, sender den besked om at den har brug for nye gener, da de gener, den har nu, ikke var nok til at sikre dens overlevelse. Den sendte besked indeholder også fitnessværdi og hvis en anden Spybot med højere fitness kommer forbi, sender den sine gener til den døde Spybot. Herefter udskifter Spybotten sine egne gener samt fitness ud med de nye gener og fortsætter med at anmode om nye gener, indtil karantæne tiden er udløbet. Herefter genfødes Spybotten med gener, der muteres. Er et individs fitness meget lav og har den ikke modtaget nye gener, genfødes den med nye gener.

Genoverførsel: Når en Spybot, der ikke er død, modtager en besked om at en anden mangler gener, går den i genoverførselstilstand, hvis modtagerens fitness vel og mærket er bedre end den dødes. I denne tilstand kan en Spybot hverken ses eller skydes af andre. Når en Spybot har sendt sine gener, skifter den adfærd tilbage til det, den var inden genoverførslen.

7.2.9 Aktuatorer

Betegner Spybottens motorer, LED'er, piezo (en simpel højtaler) samt IR transcievere.

7.3 Afrunding af implementering

Nu er der implementeret et adaptivt kontrolprogram til de autonome Spybotmodstandere, der skal indgå i robotspilsprototypen. Det adaptive element i kontrolprogrammet er baseret på en evolutionær algoritme, som er inspireret af Thiemo Krinks Motivationsnetværk. I næste kapitel skal kontrolprogrammet afprøves i praksis.

8 Afprøvning af spil

Nu hvor kontrolprogrammet er implementeret er der blot tilbage at afprøve det. Afprøvningen er ikke helt problemfri, for hvordan afgøres det om en adaptiv autonom agent udgør en fornuftig modstander i et spil ?

Der skal findes nogle objektive succeskriterier, som kan måles for at afgøre, hvorvidt de autonome modstandere tilfredsstillende opfylder deres opgave. De skal være underholdende elementer i robotspillet ved at yde passende modstand og i øvrigt opføre sig fornuftigt. Den del af robottens opgave, der går ud på at udøve modstand, kan afgøres objektivt. Underholdningsværdi er derimod ikke en direkte målbar en størrelse, så dette aspekt af robotspillet skal afprøves sammen med nogle børn i målgruppen, som kan foretage en subjektiv vurdering af de autonome modstanderes indsats.

Skal det vurderes om de autonome adaptive agenter er gode modstandere i et robotspil, kan det gøres på følgende måder:

1. En objektiv vurdering, hvor der findes et eller flere ydre målbare kriterier for, hvor godt agenten klarer sig.

2. En mere subjektiv vurdering, hvor menneskelige spillere skal afgøre, hvorvidt de autonome agenter er udfordrende og fornuftige at have til at indgå i et spil.

Nedenfor følger forskellige former for afprøvninger af det autonome agentprogram for at vurdere, om der er tale om en god robotmodstander med henblik på både punkt 1 og punkt 2.

Først vil det blive undersøgt, om de adaptive agenter kan tilpasse deres kampgener til modstandere, der konstant agerer efter den samme strategi. Her vil en gruppe af adaptive agenter blive sat til at kæmpe imod agenter med faste gener valgt med henblik på at skabe en god modstander. Resultaterne af afprøvningen vil blive understøttet af kampe mellem modstandere med andre genetiske egenskaber.

Desuden blev robotterne afprøvet sammen med nogle børn, der skal afgøre, om robotterne kan benyttes som agenter i spil. Disse spil vil komme til at foregå både i simple og mere komplekse omgivelser.

8.1 Kampscenarie: robot versus robot

Når noget skal afprøves systematisk, er det nødvendigt, at kun et aspekt afprøves af gangen. Så skal det undersøges, hvor god en robots evne til at tilpasse sig sin modstanders strategi, skal de fysiske omgivelser, modstanderens strategi, omgivelserne, lysforhold mv. være konstante.

Formålet med kampene mellem robotterne er at undersøge, om det overhovedet gør en forskel, hvorvidt generne er faste eller om de kan tilpasse sig deres modstander. Tre forskellige typer gener vil blive benyttet i afprøvningen:

1. Evolutionært tilpassede gener.
2. Faste valgte gener.
3. Faste tilfældige gener.

De evolutionært tilpassede gener er tilfældigt genererede gener, der er dynamiske og tilpasser sig modstanderens strategi over tid. Den evolutionære algoritme, der kontrollerer genernes udvikling blev gennemgået i afsnit 7.2.1.

De faste valgte gener er statiske og har alle værdien 50, der er middelværdien for generne. De formodes at være fornuftige allround kampgener, fordi det ikke umiddelbart er indlysende, om det er en fordel at være aggressiv og sej eller være mere forsigtig. Det er heller ikke nødvendigvis en fordel at være nysgerrig og med 50 som nysgerrighedsgen har robotten mulighed for at blive nysgerrig eller for at blive målrettet. Med et dovenskabsgen på 50 vil robotten bevæge sig i normalt tempo, hvilket sandsynligvis er fornuftigt, idet langsomme robotter alt for let vil blive indhentet og nedkæmpet. Omvendt kan de hurtigste robotter have problemer med at orientere sig.

De faste valgte gener er dog næppe ideelle til alle formål, fordi der ikke findes en enkelt strategi, der er ideel til alle formål. Det er netop derfor at adaptive robotter blev introduceret i robotspil. Er en bane eksempelvist meget stor og uden forhindringer, vil det nok være en fordel at kunne bevæge sig hurtigt, men samtidigt være nysgerrig for at få undersøgt banen for modstandere. Er der tale om en mindre men kompleks bane; fx en labyrint, kan det være en fordel at bevæge sig langsomt. Det kan være en fordel at være aggressiv og sej, hvis ens modstander er forsigtig eller visa versa.

Sagt med andre ord: de ideelle kampgener til alle formål findes ikke. Derimod bør der findes mindst et sæt gener, der er gode i forhold til hver fast strategi.

Med de tre nævnte typer gener er det muligt at gennemfører seks forskellige tests, som er illustreret i figur 8.1.

Gentype	Evolutionært udviklede	Faste valgte	Faste tilfældige
Evolutionært udviklede	×	×	×
Faste valgte	-	×	×
Faste tilfældige	-	-	×

TABEL 8.1: Kombinationerne af kampe mellem Spybots med forskellige gener.

Af de seks forskellige kampe, der alle ville være relevante at gennemføre, blev tre af tidsmæssige årsager udvalgt. På baggrund af de tre kampe vil det blive forsøgt afgjort, om det har en målbar effekt, hvorvidt gener er tilfældigt valgt, valgt med omtanke eller udviklet evolutionært. Kampene vil foregå mellem to grupper Spybots på hver fire individer, herefter kaldt hold α og hold β, hvor de

kæmpende grupper ikke nødvendigvis har den samme type gener. De gennemførte kampe bliver:

1. Hold α's gener bliver tilpasset evolutionært og hold β's gener er de faste valgte gener.

2. Hold α og hold β's gener er tilfældigt valgt.

3. Hold α's gener er faste og tilfældige og hold β's gener er de faste valgte gener.

Tre kampe er valgt på baggrund af forskellige overvejelser. Kamp 1 vil kunne vise om de evolutionært udviklede gener rent faktisk forbedrer sig over tid. Hvis kamp 1 ikke viser nogen klar tendens, vil man på baggrund af kamp 2 kunne afgøre, om tendensen fra 1 blot skyldes rene tilfældigheder. Kamp 3 vil kaste lys over, om de udvalgte gener rent faktisk klarer sig bedre end helt tilfældige gener.

Undersøgelserne er gennemført således, at robotterne kæmper 15 minutter af gangen, hvorefter deres gener samt antallet af nedkæmpede modstandere og egne dødsfald aflæses.

Det er nødvendigt at stoppe undersøgelsen, hver gang de data, som robotterne har opsamlet, skal aflæses, fordi Spybots ikke har mulighed for at vise mere end et tal på højest 7 bits (ved at benytte de 7 lysdioder). Derfor opsamles data fra Spybots ved at sende det til en PC via et IR tårn. Når det sker skal de andre robotter være slukkede, da forsendelsen ellers forstyrres af de andre Spybots pingpakker.

Afprøvningen vil blive udført i simple omgivelser, der er et fladt rektangulært område på 160×110 centimeter uden forhindringer. Succeskriteriet afgøres ud fra antallet af dræbte modstandere og antallet af gange robotten selv er død. Mange drab er godt og mange dødsfald er slet. Fitness burde umiddelbart være det oplagte succeskriterie, men antallet af dræbte stiger hurtigere end antallet af dødsfald og dermed stiger fitness over tid. Da Spybots med evolutionært udviklede gener kan blive genfødt flere gange i løbet af en kamp, og her tildeles start fitnessværdien 50, får disse robotter en ulempe i forhold til robotterne med faste gener.

Antallet af dræbte stiger hurtigere end antallet af dødsfald, fordi når en robot bliver nedkæmpet, så sender den besked, om at den er død og alle omkringstående modstandere får point for dette drab, idet det ikke nødvendigvis er den Spybot, der gav dødsskuddet, der alene har såret sin modstander[1]. Dermed vil antallet af drab stige hurtigere end antallet af dødsfald.

Der er ikke voldsomt mange eksempler på projekter, hvor fysiske robotter udveksler gener under udførslen af en opgave, for at tilpasse sig den opgave de skal

[1]Det kan sammenlignes med ishockey, hvor æren for et mål ikke udelukkende tilskrives den spiller, der scorer målet, men også den spiller der assisterer scoringen.

løse. Watson m.fl. gør netop det i deres arbejde, hvad de selv kalder *Embodied Evolution* [63, 23]. De afprøvninger jeg vil foretage er inspireret af de eksperimenter, der udføres i [63]. Her udveksler en række robotter gener for at blive bedre til at finde frem til et lys i midten af bane. Succeskriteriet er her hvor mange gange en robot finder lyset i løbet af et minut, uanset hvor mange gange robotten har skiftet gener undervejs.

Der er dog fundamentale forskelle mellem min og Watsons afprøvninger. Opgaven at finde lys, der hele tiden befinder sig samme sted er langt simplere og det er lettere at afgøre, om robotten har klaret sin opgave tilfredsstillende. Desuden er der ikke noget, der forhindrer robotten i at opnå sit mål. En Spybots mål, altså dens modstandere, flytter sig hele tiden og en Spybot, der forsøger at opnå sit mål bliver modarbejdet af modstanderholdet, fordi alle Spybots har som mål at nedkæmpe modstandere og samtidigt undgå selv at blive nedkæmpet.

Der er flere omstændigheder, der kan forhindre, at den evolutionære algoritme kommer til at fungere efter hensigten:

1. Fysiske omstændigheder kan være skyld i, at Spybots med gode gener klarer sig ringe.

2. Tilfældighed i kontrolprogrammet kan overskygge genernes styrke.

Af fysiske omstændigheder, som nævnes i punkt 1, er blandt andet Spybotternes tryksensor. I løbet af mine indledende undersøgelser med Spybots har jeg observeret, at trykfølerne på visse af dem er ret dårlige (de er blevet slidt). Dermed kan der opstå situationer, hvor en Spybot med gode gener, kører imod en forhindring og vedbliver med at køre frem, da trykføleren ikke registrerer noget tryk. Dermed er den kørt fast og et let offer for modstandere. Sker det flere gange vil robotten med de gode gener skifte sine gener ud, da det i kontrolprogrammet antages, at mange dødsfald skyldes dårlige gener.

I kapitel 6 viste undersøgelser, at risikoen for at IR pakker mistes, stiger i takt med at antallet af Spybots øges. Når Spybots skyder hinanden sker det ved, at de sender en *Bang du er ramt* besked til deres mål (target). Disse beskeder kan gå tabt og dermed kan en robot med gode gener bliver forhindret i at nedkæmpe sin modstander, så hurtigt som det burde være muligt.

Når en Spybot dør, sender den en *Aaaggghhh jeg er død* besked som ligeledes kan gå tabt. Dermed er det ikke sikkert, at en Spybot, der nedkæmper en anden får point for sin sejr.

Har en robot et lavt batteriniveau, vil den bevæge sig langsommere og samtidigt lader det til, at et lavt batteriniveau påvirker IR sende- og modtagerforholdene, således at Spybotten bliver dårligere til at navigere.

De nævnte fysiske omstændigheder kan få voldsom indflydelse på genernes udvikling, da forskellige fysiske omstændigheder kan have så stor indflydelse på robottens adfærd, at gode gener kommer til at virke ringe.

Punkt 2 er et problem af en helt anden karakter. Da algoritmen blev designet skulle der tages to hensyn. Robotten skulle være adaptiv og robottens adfærd sulle være uforudsigelig. For at opnå uforudsigelighed blev tilfældigheder introduceret. Dermed opstår der en lille sandsynlighed, for at en meget aggressiv og sej Spybot flygter, hvis den bliver såret. Blev algoritmen i stedet for implementeret som et motivationsnetværk [39], vil Spybottens handling altid være det samme hvis bestemte gener er over en vis tærskelværdi. Det vil give den evolutionære strategi bedre mulighed for at udvikle sig, men gøre det samtidigt lettere at forudsige Spybottens handlinger, og dermed vil den være en dårligere modstander i et spil.

Der er altså en risiko for, at tilfældighedselementet kan være med til at mindske effektiviteten af den evolutionære algoritme, men det burde ikke underminere algoritmen fuldstændigt.

8.1.1 Kamp 1

I denne kamp skal to grupper på hver fire Spybots kæmpe imod hinanden. Hold α's gener bliver udviklet evolutionært og hold β's gener er alle de faste valgte gener, der har værdien 50 gennem hele forløbet.

Forventningen til kampen er, at de valgte gener, som hold β har, vil klare sig bedst i starten. Herefter er det forventningen, at den evolutionære algoritme vil få hold α's strategi tilpasset hold β's faste strategi og dermed forbedre sine præstation i kamp.

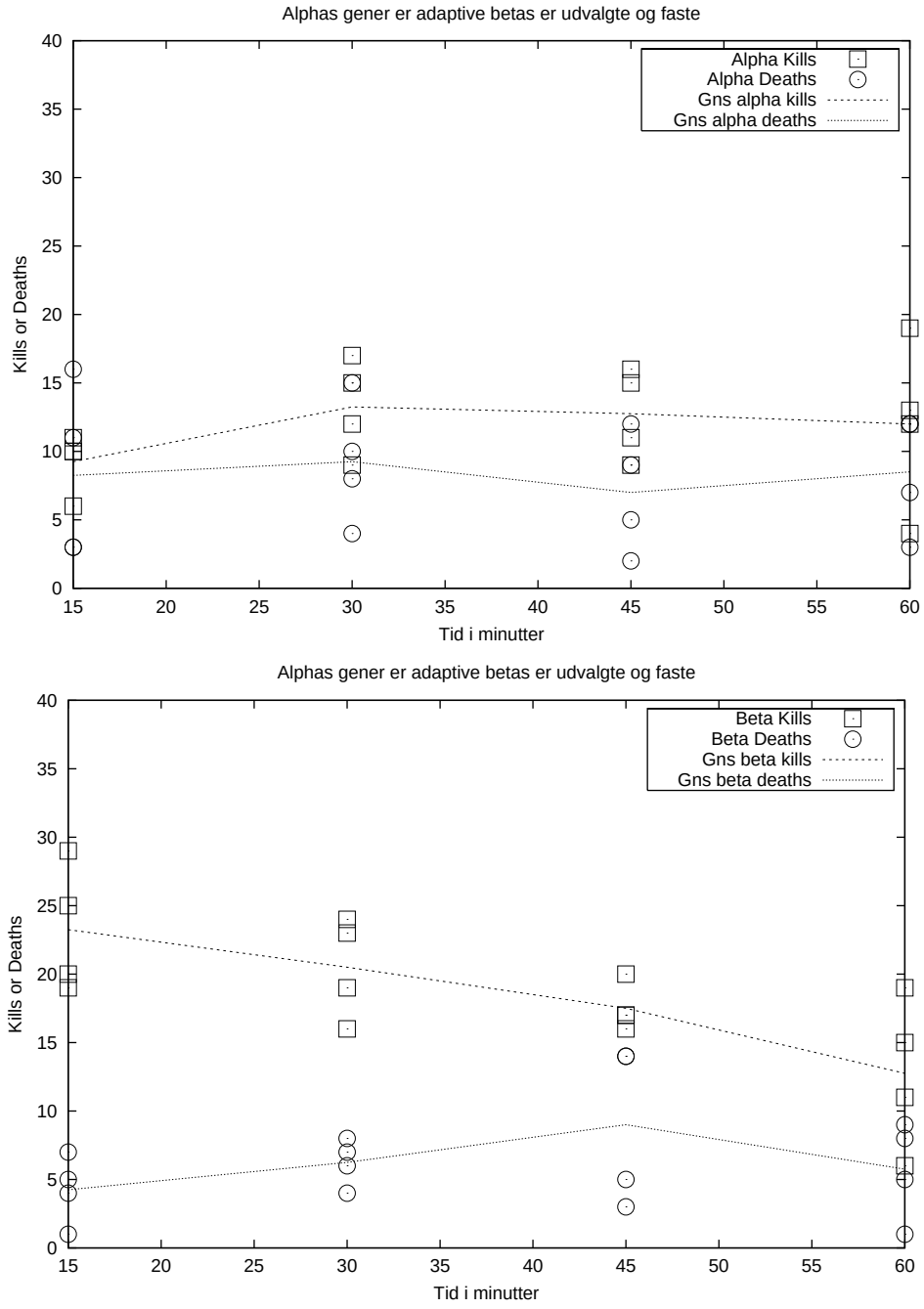
Figur 8.1 viser en kamp, af en times varighed, mellem to hold af Spybots. Øverst vises hold α, der har gener, der bliver udviklet evolutionært. Nederst vises udviklingen for hold β, der har faste gener der er valgt med henblik på at opnå gode kampegenskaber. De stiplede linjer vise middelværdien af antallet af robotter, holdene samlet har dræbt, og den prikkede linje er middelværdien af holdenes samlede dødsfald.

Det er en generel tendens for begge illustrationer, at antallet af nedkæmpede er højere end antallet af dødsfald. Denne tendens er gennemgående i alle gennemførte undersøgelser.

For hold α er der en tendens til, at antallet af nedkæmpede modstandere er stigende, når man ser på start og slutværdierne. Antallet af dødsfald er nogenlunde konstant. Tendensen for hold β, er at antallet af nedkæmpede modstandere er klart faldende og antallet af dødsfald er svagt stigende, hvis man igen sammenligner start- og slutværdierne.

Det kunne tyde på, at robotterne på hold α tilpasser deres strategi til hold β's strategi. Det skal dog også bemærkes, at hold α efter en time endnu ikke har opnået en præstation der overgår hold β's.

For at afdække om tendensen i figur 8.1 skyldes, at α's gener tilpasser sig β's strategi eller om der blot er tale om tilfældigheder, gennemføres en kamp, hvor to grupper med faste tilfældige gener kæmper imod hinanden. Viser denne kamp en tilsvarende tendens, har det ikke været en tendens, vi har set, i kampen mellem



FIGUR 8.1: Hold α med evolutionært udviklede gener kæmper imod hold beta, der har faste udvalgte gener. Hold α's tal er til øverst og β's er nederst.

de udvalgte og evolutionært udviklede gener.

8.1.2 Kamp 2

Kampen udkæmpes mellem to grupper af tilfældige gener Den generelle tendens forventes at være, at de to grupper klarer sig nogenlunde lige godt, og det forventes, at der ikke sker nogen generel udvikling af drab eller dødsfald i hverken positiv eller negativ retning.

Figur 8.2 viser, hvordan udviklingen af en kamp mellem to grupper af robotter, der begge har tilfældige faste gener ser ud. Udfaldet af kampen er nogenlunde som forventet. Det ses, at antallet af nedkæmpede modstandere og dødsfald stiger og falder efter samme mønster for både hold α øverst og hold β nederst. Det må skyldes, at på nogen tidspunkter mødes robotterne oftere end på andre tidspunkter. Umiddelbart ser det ud til, at hold β's robotter dør knap så ofte som hold α's og hold β's dræberrate er højere end hold α's i de fleste tilfælde.

8.1.3 Kamp 3

Formålet er at undersøge om de udvalgte gener er bedre end rent tilfældige gener. For at undersøge om de faste udvalgte gener, der alle har værdien 50 overhovedet er en værdig modstander, blev hold β med disse gener sat til at kæmpe imod hold α med tilfældige gener. Forventningen er at robotterne med de udvalgte gener klarer sig bedre end robotterne med de tilfældige gener.

Forløbet af denne kamp er illustreret i figur 8.3. Hold α, hvis udvikling, der er illustreret øverst, har faste tilfældige gener. Udviklingen for hold β, der har faste udvalgte gener, er illustreret nederst.

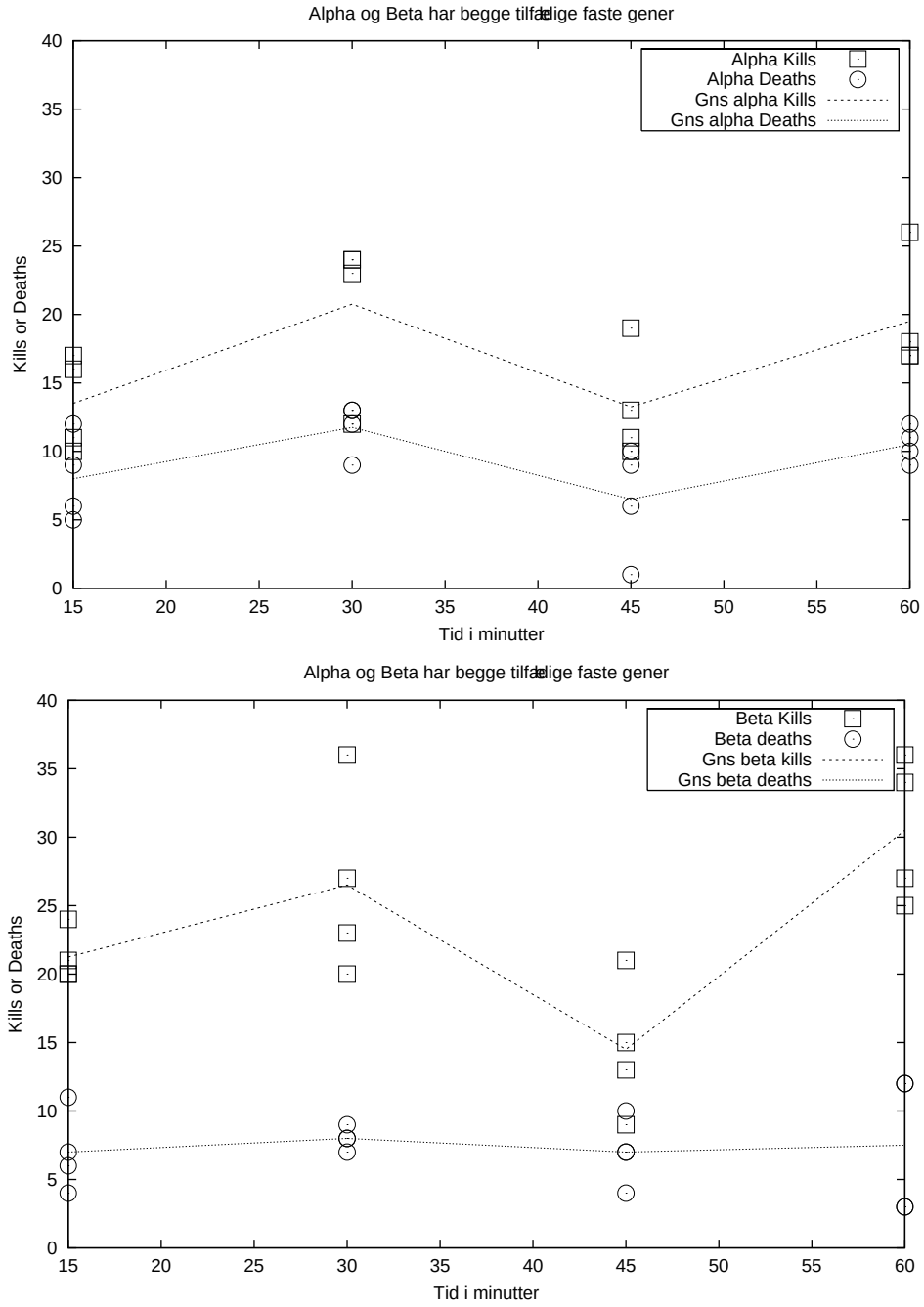
Antallet af dødsfald er højere for hold α end for hold β, og antallet af dræbte er højere for hold β end for hold α. Dermed kan det fastslås, at hold β med de udvalgte gener klarer sig bedre end en gruppe Spybots med tilfældige gener.

Det bemærkes, at antallet af nedkæmpede modstandere er højere for hold α med det tilfældige gener i figur 8.3, i forhold til hold α i figur 8.1. Omvendt bliver robotterne fra hold α med det tilfældige gener dræbt oftere end deres fæller med evolutionært udviklede gener.

8.2 Opsamling af resultaterne fra kamp 1-3

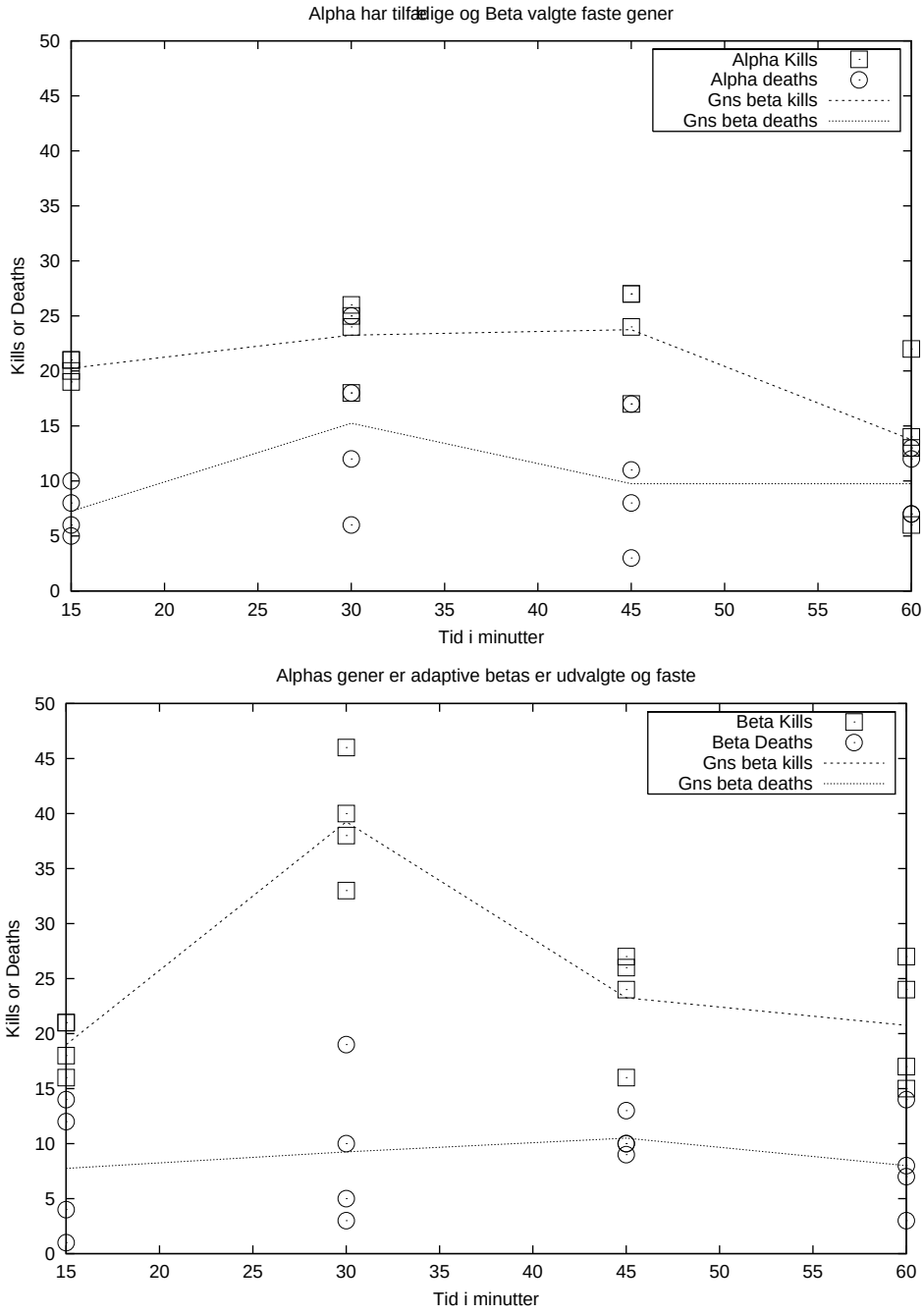
Jeg har påvist, at det er muligt at implementere en evolutionær algoritme i et robotspil, hvor Spybots er de autonome modstandere på trods af, at der er mange omstændigheder, der kunne have forhindret, at der fremkom fornuftige resultater.

Figur 8.1, 8.2 og 8.3 giver et samlet indtryk af, at robotterne med de evolutionært udviklede gener forbedrer sig over tid. Det er muligt, at tilpasningen tager for lang tid til, at det kan benyttes i en robotspil men det vises dog, at ideen om at benytte adaptive robotter baseret på en evolutionær algoritme i robotspil, er brugbar, omend der skal flere og mere dybdegående undersøgelser til for at forbedre resultaterne for de autonome adaptive robotter.



FIGUR 8.2: To grupper af robotter med tilfældige gener kæmper imod hinanden. Hold α's tal er til øverst og β's er nederst.

De faste udvalgte gener klarer sig bedre end de tilfældige og de evolutionære. Det vil sige, at det ikke er ligegyldigt, hvilke gener en robot har. Det er højest sandsynligt, at den evolutionære algoritme kunne justeres bedre, end jeg har



FIGUR 8.3: Hold α med tilfældige gener kæmper imod hold β med udvalgte gener. Hold α's tal er til øverst og β's er nederst.

fået gjort. Hvis strategien for overkrydsning og mutationen bliver ændret, kan det sagtens tænkes, at den evolutionære algoritme ville finde et bedre sæt gener hurtigere, end det er tilfældet på nuværende tidspunkt.

Det ville måske være nyttigt at ændre gener oftere, end når robotterne dør. Det ville også være interessant at undersøgt hvordan gener udvikler sig, hvis tilfældighedselementet i kontrolprogrammet i forbindelse med valg af adfærd blev elimineret.

Generelt ville det være rart at have flere målepunkter og/eller lade robotterne køre længere tid. Desværre er det omstændeligt og tidskrævende process, at overføre data fra Spybots til en PC. For ikke at sprænge den tidsramme der var afsat til afprøvningerne i for høj grad valgte jeg at benytte færre målepunkter.

Desuden kunne det være interessant at gennemføre kampe mellem robotter med evolutionære gener og robotter med tilfældige gener og lade to grupper, af robotter med de udvalgte gener kæmpe imod hinanden. En kamp mellem to grupper der begge har evolutionære gener ville ligeledes være interessant at se resultatet af. Det kunne også være interessant at variere de fysiske omgivelser og se, om det giver anderledes resultater.

Hver test burde være gennemført mere end en gang for at sikre større sikkerhed omkring resultaterne.

8.3 Afprøvning sammen med børn

Efter den objektive undersøgelse er det tid til en mere subjektiv vurdering af hvorvidt kontrolprogrammet er velfungerende. Det er ingenlunde givet, at en robot med det, som i afsnit 8.1, blev betragtet som gode kampgener, bliver en fornuftig modstander i et robotspil.

Inden afprøvningen havde jeg besluttet, at de autonome Spybots skulle afprøves på en række forskellig måder. De skulle afprøves i simple og i mere komplekse omgivelser. De autonome Spybots skulle agere både modstandere og medspillere til de menneskelige spillere.

De simple omgivelser skulle være en arena omgivet af en mur som i de ovenstående undersøgelser. De komplekse omgivelser kunne dels være inspireret af de Spybot spil der blev beskrevet i 5.6 og dels blive opfundet af børnene i løbet af evalueringen af spillet.

Til at hjælpe med at afprøve de forskellige robotspil fik jeg hjælp af Martin og Sebastian, der begge er 10 år og går i tredje klasse på Århus Friskole. De var forbi mit kontor torsdag og fredag i uge 17; begge dage fra 9 til 12. Dermed var der mulighed for omprogrammering af Spybots mellem vores møder. Desuden kunne tiden mellem vores møder benyttes til opbygning af baner som vi ikke havde til til at konstruere sammen.

Det forventes, at Spybottens tryksensor kan blive et problem, når de skal kæmpe imod menneskelige spillere. En hjælpeløs robot, der står stille og kører imod en forhindring er et let offer, der måske ville blive overset af en autonom spiller men næppe af en menneskelig.

I kampscenarier, hvor to spillere kæmper imod en hær af robotter forventes det ligeledes, at spillerne har en stor fordel idet de kan koordinere deres strategier for

at opnå deres mål. Desuden har de personer, der spiller robotspil med Spybots som aktører, den fordel, at de har global information om spillet idet de kan overskue hele spilområdet.

I forlængelse af disse konstateringer skal det dog huskes, at målet ikke er at skabe Spybot modstandere, der til enhver tid kan nedkæmpe de spillerne, men autonome modstandere, der yder modstand nok, til at det er underholdende at spille imod dem.

For at udligne spillernes overlegenhed en smule, blev de menneskestyrede Spybots udskiftet med en model, der ikke kunne køre nær så hurtigt som de autonome. Desuden bliver spillerne ikke genoplivet, når de først er døde, hvorimod de autonome Spybots blot har karantæne i 30 sekunder, når de dør.

8.3.1 Robotspil i simple omgivelser

Første robotspil havde jeg på forhånd forberedt; det gik i alt sin enkelthed ud på, at de to drenges Spybots skulle kæmpe imod mindst fire autonome robotter i den åbne arena, der også blev benyttet til ovenstående undersøgelser. I starten forløb spillet som forventet, drengene kæmpede og var overlegne, men blev ind i mellem overvundet af de autonome Spybots. Figur 8.4 viser en kamp mellem mennesker og maskiner. Den viser også en af fordelene ved robotspil frem for de ordinære computerspil - det er muligt at interagere direkte med aktørerne, hvis de ikke gebærder som de skal.

Spillet var helt åbenlyst alt for simpelt, så drengene begyndte at gå i nærkamp med de autonome Spybots ved at køre ind i dem. Reglerne blev udvidet til, at hvis en robot blev væltet, så den lå hjælpeløst med larvefødderne i vejret var den død, desuden var der stadig mulighed for at skyde og blive skudt. Døde Spybots skulle straks fjernes fra banen. Drengenes nye nærkampsstrategi havde den ulempe, at de oftere blev dræbt og det endte med, at de deaktiverede spillerprogrammet, så de hverken kunne skyde eller blive skudt og nøjedes med at foretage brydekampe som de naturligvis vandt, da de autonome Spybots ikke var forberedt på nærkamp med udødelige modstandere. Figur 8.5 viser nærkampe mellem aktanter og agenter.

Dernæst prøvede vi en anden type spil, hvor drengene havde hver en gruppe medhjælpere på fire Spybots og målet var nu at udrydde hinandens grupper. Endnu en gang udviklede spillet sig til en brydekamp, denne gang mellem de to drenge, imens de autonome Spybots bekæmpede modstanderens hold.

Alt i alt virkede de autonome Spybots som forventet, de ydede en vis modstand og deres angreb kom til at virke koordinerede, idet fire autonome Spybots, der kæmper imod to aktør Spybots inden for et begrænset område, ofte vil vælge det samme mål, som de vil angribe samtidigt. Da robotterne ikke kommunikerer eksplicit for at koordinere deres angreb er de "koordinerede" angreb et emergent ved træk ved robotspillet.

FIGUR 8.4: Kamp mellem menneskestyrede og autonome Spybots i simple omgivelser. Sebastian der var blevet trængt op i en krog giver sin Spybot en mere gunstig placering.

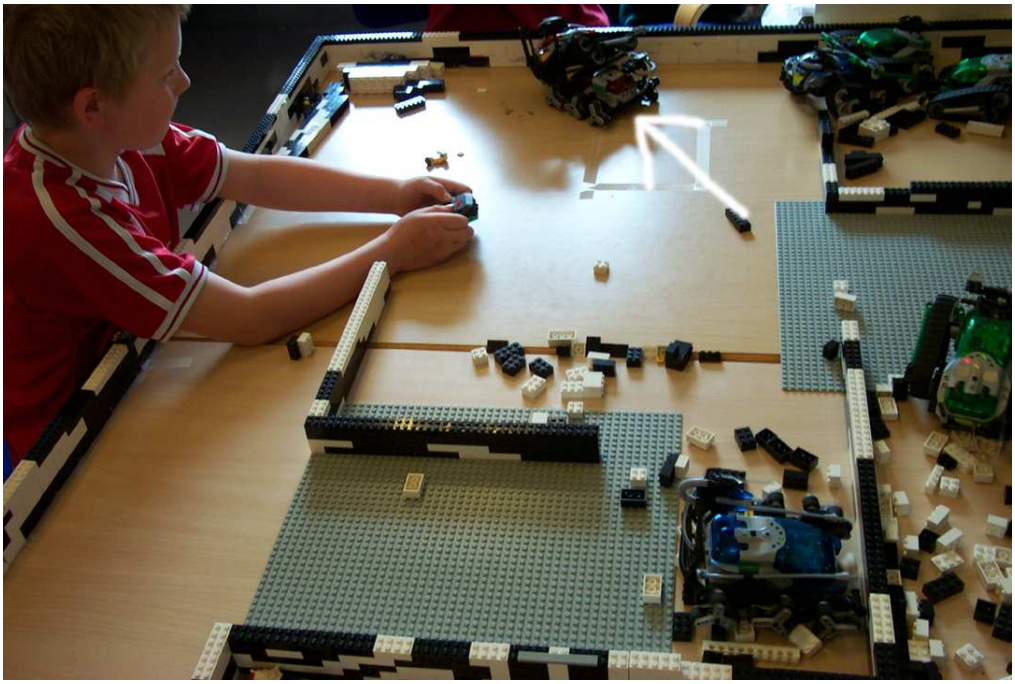
8.3.2 Robotspil i komplekse omgivelser

For at gøre spillet mere udfordrende for både drengene og de autonome Spybots blev der tilføjet forhindringer til banen. De bestod af alt lige fra mure bygget af LEGO og løse LEGO klodser spredt ud over banen, til tunge bøger og hvad der nu ellers var inden for rækkevidde. Figur 8.6 viser Spybots i mere komplekse omgivelser. Til tider var LEGO mænd en del af opstillingen især *Borgmesteren*, en LEGO mand med høj hat spillede en overgang en vis rolle i vores robotspil. Borgmesteren sad fx på Sebastians Spybot og "fortalte" ham hvilken modstander han skulle angribe eller hvilken vej han skulle køre.

De relativt få ændringer på banen gav et helt andet spil, eller en helt anden leg er måske en bedre betegnelse. Spil er som tidligere nævnt bundet af faste regler hvorimod legens regler opstår spontant og kan ændres under forløbet. Mens der blev spillet robotspil gik spillet hen og blev leg i og med at reglerne for spillet heletiden blev ændret.

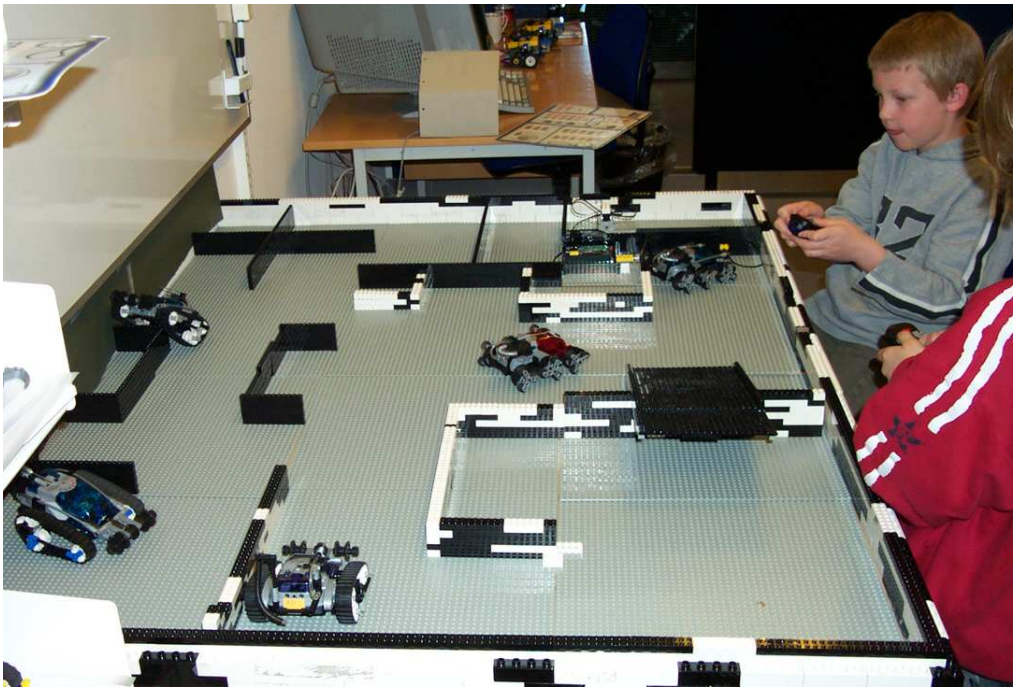
Til tider blev spillet, gennemført som jeg oprindeligt havde tænkt mig, nemlig som en form for computerspil i det fysiske rum med robotter som aktører. Her klarede de autonome agenter sig udemærket, hvilket vil sige, at de ydede en vis

FIGUR 8.5: Brydekamp. Sebastians robot (markeret med *S*) bliver overfaldet af en række autonome Spybots. Martins robot (markeret med *M*) iler til undsætning.



FIGUR 8.6: Kamp i komplekse omgivelser uden bøger. Den hvide pil peger på Sebastians robot.

modstand, men de virkede på ingen måde overlegne. De havde flere problemer end i de første forsøg, fordi de ikke opdager forhindringer, før de støder ind i dem. Havde Spybotten været udstyret med IR sensorer, der kan benyttes som afstandsmålere, som fx *Khepara* eller *AIBO* robotterne er udstyret med, ville navigation i labyrintlignende omgivelser ikke være nær så stort et problem.



FIGUR 8.7: Labyrint til robotterne.

Da drengene mødte op på mit kontor for anden gang havde jeg forberedt nogle labyrint elementer bestående dels af passive forhindringer som mure og dels aktive forhindringer som var en vippeplade og en automatisk dør styret af en *CodePilot*. Forhindringerne var lig de, der blev beskrevet i afsnit 5.6.

Sammen fik vi de forberedte forhindringer, samt nogle som blev fremstillet på stedet bygget sammen til en labyrint, se figur 8.7, som skulle afprøves sammen med de autonome Spybots. Banen var, i hvert fald i starten, sjovere for drengene, mens de autonome Spybots havde problemer med at komme rundt i labyrinten. Og derfor endte denne afprøvning med, at de to drenges Spybots sneg sig rundt i labyrinten og nedkæmpede deres næsten forsvarsløse modstandere. Det var ingen krævende opgave for drengene, men det hjalp alligevel på selvtilliden, at de to havde nedkæmpet ikke mindre end otte dræberrobotter.

8.4 Opsamling af erfaringer af spiltesten

Robotspils største fordel er formentligt, at det i modsætning til de klassiske computerspil er et fleksibelt interaktivt spil. Spilleren eller spillerne kan bygge baner op som de har lyst til i de materialer man lige har ved hånden. Spillets regler er hele tiden til forhandling, hvilket ikke er tilfældet i de computerspil, som ellers spilles af børn. Modstanderen har godt nok en fast opgave, men det er alligevel muligt at modellere mange typer spil eller lege med den samme modstander.

Desuden kan robotspil foregå, hvor børns lege normalt foregår, i børneværelset, i stuen, i frikvarteret i skolen eller måske endda i haven. Computerspil derimod er ofte noget, der skal foregå stillesiddende hjemme foran PC skærmen.

Om robotspil af den type, jeg har eksperimenteret med snarere er robotunderstøttet leg, kan diskuteres, men de kan være et ganske underholdende supplement til de mange interaktive computerspil, der allerede findes. Drengenes tilgang til spillet minder om det Klaus Testrup beskriver som **Medielege** [62] som han beskriver således:

> Medieleg er, når børn selv bruger medierne som råstof i deres legekultur. Legen kan foregå med teknologien selv. Børnene bruger et kamera, en mobiltelefon eller et andet stykke teknik som redskab i legen.

Citatet udtrykker meget godt, hvad der skete under afprøvningen. Jeg præsenterede et udkast til et robotspil og under afprøvningen gik robotspillet fra blot at være et spil til at være en hybrid mellem robotspil, konstruktionsleg med LEGO klodser og rollelege med LEGO mændene. Fænomenet var tilsvarende, da Klaus Testrup og jeg arbejdede med robotter i 0.B og i *Robotterne går sig en tur* [36] af Testrup, Caprani m.fl. bliver tilsvarende observationer beskrevet.

Jeg vil vove den påstand, at robotspil, som de, der blev gennemført med Martin og Sebastian, møder børn på deres egne præmisser i langt højere grad end de klassiske computerspil gør.

Alt i alt efterlod afprøvningen af Spybots som modstandere i et robotspil mig med det indtryk, at ideen om robotspil er værd at bygge videre på, selv om det nok ikke er Spybots, som spillet skal være baseret på, da de har for mange mangler med hensyn til at skabe sig et overblik over verden. Desuden ville mulighed for bedre lyd og et display være rart i forbindelse med at skulle designe og afprøve robotspil.

9 Konklusion og perspektivering

Hovedkonklusionen for specialet er, at det er muligt at implementere sjove, udfordrende og underholdende robotspil. Rent datalogisk er det absolut ikke en triviel udfordring at udvikle de autonome agenter, der skal indgå i et sådant spil, fordi robotterne skal agere i komplekse og uforudsigelige omgivelser.

Undervejs i forløbet er udviklingen af et robotspil for børn i alderen 9-13 år, fra ide til prototype blevet beskrevet. Det er på baggrund af mine undersøgelser blevet påvist, at det er muligt at konstruere underholdende robotspil, inspireret af computerspil, for børn i alderen 9-13 år. Det spil, der er blevet udformet, er et arkadeinspireret skydespil og det har ingen rammefortælling.

At robotspillet er fysisk gør, at spillets regler hele tiden kan forhandles og udvikles. Dermed kommer det at spille robotspil til at minde mere om en leg eller medieleg [62], end om et spil. Robotspil møder børnene på deres egne præmisser i højere grad end computerspil gør, netop fordi børnene kan tilføje elementer og begreber fra deres egen legekultur til robotspil.

Der viste sig også nogle helt fundamentale forskelle mellem computerspil og robotspil. Spilleren eller spillerne har mulighed for at interagere direkte med spillets aktører og omgivelser, hvilket giver langt større mulighed for variation, men omvendt kan det betragtes som en form for snyd, idet spillets regler kan omgås. Robotspillets brugere har mulighed for selv at opbygge og ændre de omgivelser, som spillet foregår i, hvilket er betydeligt vanskeligere i traditionelle computerspil. Der findes computerspil, der tilbyder mulighed for at opbygge egne baner. Men banerne skal konstrueres ud fra bestemte principper og reglerne for det spil, der skal udspille sig på banen er stadig fastlåste og er ikke til forhandling.

Det er sværere at implementere agenter til et robotspil end til computerspil, fordi der kan opstå en række udfordringer på grund af fysiske omstændigheder, som man ikke vil støde på i computerspil. I løbet af dette projekt har nogle af udfordringerne været, at der opstår pakketab, når flere Spybots er samlet, nogle trykfølere var dårlige, Spybots vælter og er dermed ude af stand til at løse deres opgave og Spybots har ikke sikker global information. De virtuelle modstandere i spil agere under lettere vilkår, da omgivelserne her kan kontrolleres fuldstændigt af programmøren.

Det blev afgjort, at en robotmodstander bør være adaptiv for at kunne tilpasse sig spillerens strategi. For at opnå det, blev der benyttet en evolutionær algoritme inspireret af Thiemo Krinks *motivations netværk* [39]. Effekten af den evolutionære algoritme var ikke overbevisende, men robotterne, hvis gener blev udviklet, forbedrede sig en smule i løbet af en time. Det er ganske givet, at hvis der blev lagt flere kræfter i at justere parametrene i den evolutionære algoritme, vil der kunne opnås bedre resultater. Brugen af en simulator til at udvikle et kontrolprogram og/eller udvikle Spybots gener blev fravalgt, da det blev vurderet, at den tid, det ville tage at udvikle en tilpas god simulator, ville være bedre brugt på at håndkode et kontrolprogram og foretage nogle fysiske undersøgelser. Det

skal bemærkes, at kontrolprogrammet til en robotmodstander kan designes på et utal af måder. På baggrund af en analyse har jeg valgt en strategi, som synes at være fornuftig, men dermed er det ikke afgjort, at det ikke kan gøres bedre.

Spybotten er ikke den mest velegnede platform til at implementere et robotspil, for der er for få sensorer til at give et fornuftigt indtryk af omgivelserne. Især i komplekse omgivelser som labyrinter eller labyrintlignende omgivelser er det vanskeligt at få Spybotten til at navigere sikkert rundt, da den kun er udstyret med en lys- og en tryksensor. Desuden får Spybotten problemer med at orientere sig i forhold til andre Spybots, når de optræder i større antal, da de pakker de hele tiden sender, forstyrrer hinanden. Spybots har dog en klar fordel ved deres robusthed for de er blevet udviklet til at børn skal lege med den. Adskillige gange er Spybots blevet tabt på gulvet uden at tage skade, det er knap så sikkert, at robotter som *Khepera* og *AIBO* ville kunne klare den behandling. Spybotten har ligeledes den fordel at den koster under en tyvendedel af *Khepera* og *AIBO*. Dermed er det realistisk at robotspil baseret på teknologi inspireret af Spybots vil kunne sættes i produktion og købes af andre end millionære.

Da arbejdet med udviklingen af adaptive autonome agenter startede, gik arbejdet langsomt i første omgang, fordi jeg ikke kunne skaffe det fornødne udviklingsværktøj. Undervejs i udviklingsprocessen blev arbejdet vanskeliggjort af at Spybottens firmware og *Spybot Engine*, som modstander robotternes kontrolprogram bygger ovenpå, til tider opfører sig uforudsigeligt. I starten af denne proces blev Michael Barratt Andersen fra LEGO Virtual i nødstilfælde benyttet som rådgiver - sidenhen blev han desværre sparet væk.

Debugging af Spybots var også vanskelig, fordi kontrolprogrammer skulle debugges ved hjælp af 8 lysdioder og en piezo højtaler. Senere fandt jeg en måde at benytte infrarøde beskeder til debugging, hvilket lettede denne del af arbejdet, selv om det aldrig blev så let som, når man skriver applikationer til en PC i sprog som *C* og *JAVA*.

De næste afsnit skal give et indtryk af i hvilken retning, jeg kunne tænke mig at mit arbejde skulle gå hvis jeg havde 3-4 måneder yderligere til at arbejde målrettet med robotspil.

Hvis der havde været mere tid til rådighed kunne den med fordel benyttes til nærmere undersøgelser af den evolutionære algoritme med henblik på at justere parametre, således at de autonome adaptive robotter kan blive i stand til at klare sig endnu bedre. Samtidigt ville det være interessant at fjerne tilfældighedselementet fra kontrolprogrammet for at undersøge, hvordan det vil påvirke henholdsvis den evolutionære algoritme og robotternes adfærd.

De kampe, der blev gennemført i afsnit 8.1, burde være gennemført flere gange for at eliminere risikoen for fejlmålinger og de kampe, der ikke blev gennemført, skulle udkæmpes. Det ville give en bedre indikation, af hvilken type gener der klarer sig bedst.

Det vil være en spændende opgave at skabe et robotspil hvor æstetiske elementer fx fortællingen, blev indarbejdet i højere grad. Belønningsdelen bør der

også arbejdes mere med. Ganske vist er det en belønning i sig selv at nedkæmpe en modstander, men hvis point blev indført kunne spilleren måle om han har forbedret sin præstation siden sidste spil. Desuden får han mulighed for at prale over for sine venner hvis han er dygtig. Forholdet mellem teknikken og æstetikken i spil hænger nært sammen med spiloplevelsen så det er bestemt et område der bør undersøges yderligere. Vil man udvikle et salgbart produkt er det ganske givet at en veludviklet æstetiske dimension er nødvendig.

Det vil også være nyttigt at få nedskrevet ideer til robotspil. Lige fra de store og vilde til de mere realistiske, for dermed at få et bedre overblik over robotspils fremtidige muligheder, såfremt nogen skulle finde på at arbejde videre med dem. Hvis der er et marked for robotspil, hersker der ingen tvivl om, at nogen nok skal tage arbejdet med robotspil op.

Det mest ideelle ville være at udvikle robotter specielt til robotspil og overveje hvordan de bedst kom til at passe sammen med børns øvrige lege og medielege. Det ville måske være naturligt at lade børns mobiltelefoner blive en del af robotspil. Robotten kunne styres via bluetooth kommunikation og kunne sende SMS eller MMS, beskeder når den stødte på udfordringer eller problemer. Måske skulle robot og spiller kunne kommunikere ved at "tale" sammen gennem mobiltelefonen.

LEGOs konstruktionsprincip kunne med fordel benyttes til at lade spillerne konstruere deres robotter af moduler med forskellige egenskaber. Fx kunne der være mulighed for at udstyre sin robot med forskellige våben, hjul og sensorer, der hver især havde forskellige egenskaber. Dermed ville der kunne opnås større variation i spillet eller legen, om man vil.

Alt i alt har arbejdet med robotspil overbevist mig om, at her er et område, hvor der er et enormt potentiale for at skabe underholdende spil. Samtidigt er der store datalogiske udfordringer i at få dem til at fungere. Det spil, jeg har arbejdet med er kun toppen af isbjerget, men det giver alligevel et indtryk af de muligheder, robotspil tilbyder, som andre spiltyper ikke gør.

10 English Summery

During the last years the price on robots has dropped dramatically. So today it is possible to buy robots equipped with communication equipment. Robots of this kind were only available to researchers at the Universities and in the Army until a few years ago. This cheaper technology opens new possibilities for the gaming industry because it makes it possible to create interactive games, which does not take place on a screen but in the physical world. In this master thesis I try to develop a prototype for an entertaining interactive robot game for children in the age 9 to 13. The game will be based on the commercial robot toy product Spybotics, which is produced by LEGO. During the process I try to answer these questions in order to succeed.

- How should a funny and challenging robot game for children in the age 9 to 13 look?
- Is it possible to transfer elements from ordinary computer games to robot games?
- What kind of control program should the opponent robot have?
- Should the control program be implemented as an adaptive autonomous agent?
- What kind of commercial product can be used in a robot game, when both technical issues and price should be taken into account?

To find the key to an entertaining and challenging robot game, traditional computer games will be investigated to find elements, which may be used in robot games. Afterwards agents in general will be investigated in order to find a suitable strategy for the control program for the autonomous robots, which should be opponents in the robot game prototype. Then the control program will be implemented and the game will be tested with some children in the target group for the game.

The main conclusion is that it is possible to create a robot game where the opponents are autonomous Spybots. It is to a certain degree possible to make the opponents adaptive, so the will be better opponents, and the game will be more challenging. During the process it is discovered that the simple Spybot robot works fairly in the robot game, but the number and quality the Spybots sensors are too limited to make a strong opponent. Sony's AIBO robot and K-Team's Khepera II robot are suggested as alternatives. They will probably perform better in a robot game however they cost more than twenty times the price of the Spybot, so this solution is rejected. The perfect robot should be made for robot games but with better sensors than the Spybot. An interesting observation when testing

the robot game with children was that the activity to play an interactive robot game, is closer to play a children's games than to play computer games.

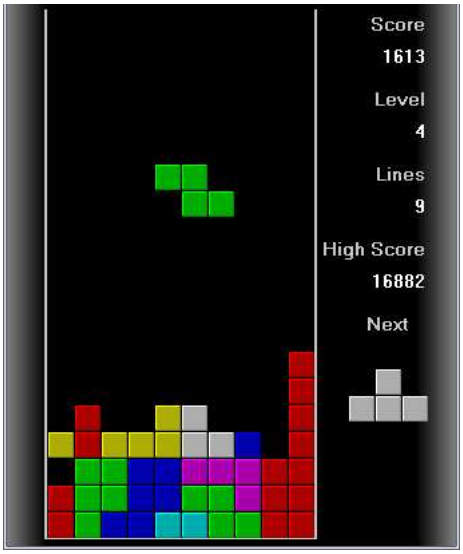
This project has convinced me that interactive entertaining robot games are an area which should be discovered further.

A Spilbeskrivelser

Informationer om de forskellige spil er hentet fra *Den digitale leg* [55]og *The Ultimate History of Video Games* [35]. Desuden er der hentet billeder og informationer fra internettet.

A.1 Tetris

I Tetris skal spilleren vende og flytte geometriske former der falder ned fra toppen af skærmbilledet. Formerne fylder alle tern og spilleren skal få dem til at passe sammen. Når en række på langs er fyldt med tern, forsvinder alle ternene og dermed bliver der plads til flere faldende figure. Spilleren taber når formerne bliver stablet så højt at de når til skærmbilledets top.



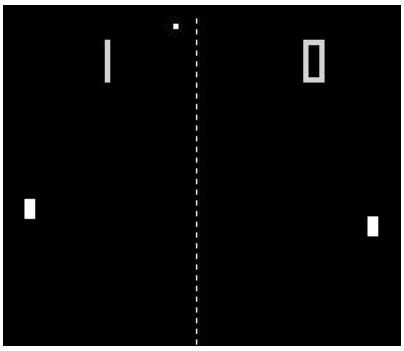
FIGUR A.1: Screenshot af det klassiske arkadespil *Tetris*

Tetris lyder måske ikke særligt sjovt, men det er et udfordrende og vanedannende spil. Et screenshot fra Tetris kan ses i figur A.1.

A.2 PONG

Dette spil produceret af ATARI i 1972, er en af de første store komercielle successer. Spillet er inspireret af bordtennis og går ud på at skyde "bolden" over på modstanderens banehalvdel ved hjælp af sit "bat". PONG var et af de allerførste spil der var tilgængeligt i spillearkaderne.

Et screenshot fra spillet PONG kan ses i figur A.2.



FIGUR A.2: Screenshot af det klassiske arkadespil *Pong*

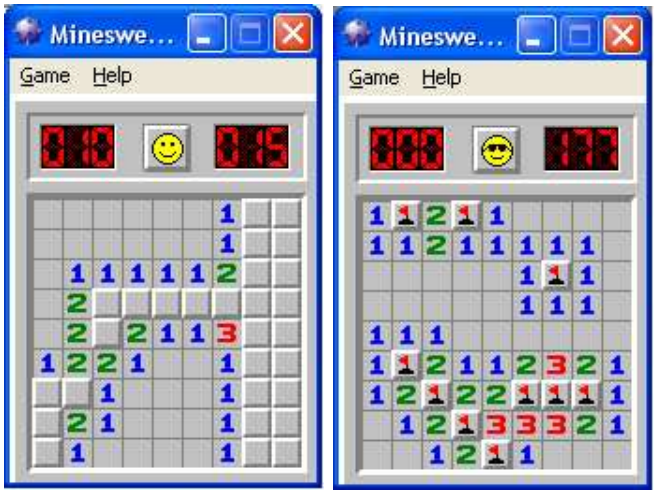
A.3 Pacman

Pacman udviklet Midway i 1981. Spillet går ud på at føre den gule ost gennem labyrinten og spise alle prikkerne. Fire spøgelser forsøger at forhindre spilleren i at nå sit mål. Disse kan dog spises når Pacman spiser en af de fire energipiller der befinder sig i labyrinten. Pacman var et af 80'ernes største arkadespilhits og det var muligt købe alt lige fra Pacman tøj til Pacman gulvtæpper og gardiner.

FIGUR A.3: Screenshot af det klassiske arkadespil *Pacman*

A.4 Minestryger

Minestryger der er udviklet og genudviklet af Microsoft 1981-03, er et strategispil hvor spilleren har brug for logisk tænkning og hurtighed. Spillern skal klikke på felter, hvor han regner med der ikke er miner, er der ikke det står der ud rundt om feltet, hvor mange miner feltet grænser op til. Klikker man på en mine har man tabt og spillet vindes når der kun er miner tilbage.



FIGUR A.4: Screenshots fra tidsrøveren *Minestryger*

Screenshots fra minsesryger kan ses i figur A.4

A.5 Space Invaders

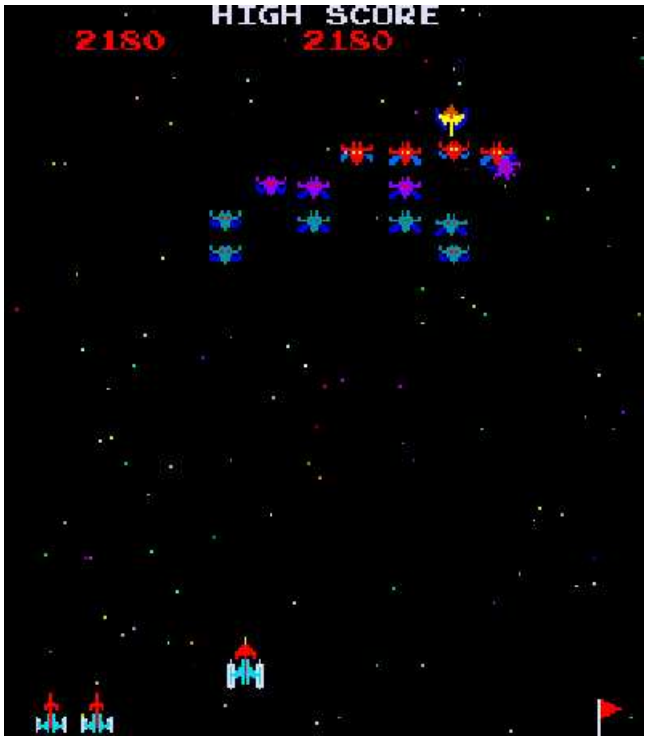
Space Invaders er et simpelt arkadespil og var en af de første computerspil der blev en kassesuccess. Et screenhot fra spillet ses på figur A.5. Handlingen er kort at en inverderende hær fra rummet, monstrene i toppen skal skydes ned af dig som styrer raketten i bunden af billedet.

A.6 Starcraft

Starcraft er et realtime strategispil der blev udgivet i 1998 af Blizzard Ent. Starcraft udspiller sig i rummet og går ud på at samle recourcer, opbygge hære og bekæmpe aliens. Starcraft kan spilles af op til 8 spillere. Mere info kan findes på [1].

A.7 Tomb Raider

Er en blanding mellem tredjepersons skydespil og eventyrspil. Det er udviklet af Eidos Interactive og blev udgivet i juli 2003. Spilleren styrer hovedpersonen Laura



FIGUR A.5: screenshot fra spillet Space Invaders (Taito 1978)

FIGUR A.6: Screenshot af space strategi spillet *Starcraft* Blizzard Ent. 1998. Billedet er hentet fra [9]

Croft gennem en række opgaver, hvor hun er nødt til at nedkæmpe forbrydere og fæle væsner.

Screenshots fra spillet Tombraider - Angle of Darkness, kan ses på figur A.7.

FIGUR A.7: Screenshot fra eventyrspillet *Tombraider - Angle of Darkness*. Billedet er taget fra [6]

A.8 Civilazation II

Civilazation II er udviklet af Microprose 1996. Det er et turbaseret strategispil, der dels går ud på at skabe en højt civiliceret befolkning, dels at opbygge hærenheder og bekæmpe sine modspillere. Spillet kan vindes på syv forskellige måder; fx have verdensherredømmet, eller være den første nation, der sender folk ud i rummet.

FIGUR A.8: screenshot fra spillet *Civilazation II*

Screenshots fra Civilazation II kan ses i figur A.8.

A.9 Splinter Cell

Tredje persons skydespil, nærmere betegnet af typen **stealth** som er actionspil hvor hovedpersonen Sam Fisher en stor del af tiden skal snige sig frem og overmande sine modstandere. I spillet styre man Sam Fisher en amerikansk elitesoldat

der skal forhindre en tjetjensk terrorist i at bruge sit hemmelige våben *The Ark* mod mål i USA.

FIGUR A.9: screenshot fra spillet *Splinter Cell* der satte nye standarter for grafik i skydespil

Spillet er produceret af Ubi Soft i 2002 og screenshots kan ses i figur A.9.

A.10 Doom II

Doom II er et actionskydespil der er udviklet af ID soft i 1995. I Doom II bliver spilleren briefet om, at væsner fra helvede er begyndt at dukke op og de skal bekæmpes. Spillet afsluttes, når spilleren overvinder alle monstre eller at de fæle monstre får bugt med spilleren.

Spillet har en multiplayerdel der gør det muligt at spille flere personer imod hinanden over netværk.

Screenshot fra spillet Doom II kan ses i figur A.10.

A.11 Quake II

Endnu et hæsblæsende 3D skydespil. Spillet er udviklet af ID Soft i 1996. Spilleren bevæger sig rundt i labyrinter og bekæmper monstre. Quake II har lige som Doom II en god multiplayer del.

Efterfølgeren Quake III har en multiplayer del der er så god at der er blevet afholdt verdensmesterskaber i Quake III og der er enkelte personer der lever fuldtids af af spille Quake III.

Screenshot fra spillet Quake II kan ses i figur A.11.

A.12 Wolfenstein 3D

Wolfenstein er et action/arkade produceret Westwood Studios 1993. Spillet var det første 3D skydespil og grundlagde en helt ny genre inden for computerspillene.

FIGUR A.10: screenshot fra spillet *Doom*

FIGUR A.11: screenshot fra spillet *Quake*

Spilleren styrere spillets hovedperson B. K. Blazkowicz, der er på jagt efter nazisternes hemmelige våben på slottet Wolfenstein. Undervejs skal han nedkæmpe nazister og samle deres skatte der ofte er gæmt bag hemmelige døre.

Screenshots fra spillet Wolfenstein 3D kan findes i figur A.12.

FIGUR A.12: screenshot fra spillet den absolutte klasikker inden for 3D skydespil; *Wolfenstein 3D*

A.13 Pixeline

I Pixeline 1 - Syng og leg, kan du hjælpe Pixeline med at finde sine aber ude i junglen, eller du kan tage med til hendes mors lille dyrehandel og lege med dyrene.

I slikbutikken gælder det om at være hurtig og fange de fem tossede vingummifrøer, og ude på landet skal man hjælpe den gamle sky med at vande både blomster og marker og træer. Der er også nogle skøre elefanter, som du kan få ud at gå på edderkoppespindet i mit cirkus, og min ven buschaufføren tager os med på den ene tur efter den anden.

Teksten er uddrag fra producentens hjemmeside [4]

B CDROM

CDROM'en indeholder:

Programmet for den autonome modstaderrobot og for spillerrobotten.
Billeder fra spilafprøvningen med Martin og Sebastian
Spybotics ROM Dokumentationen.
ApplikationenLEGOMindstormsSDK25.exe, der indsterllerer ScriptEd

Litteratur

[1] Blizzard ent. side om starcraft. www.blizzard.com/starcraft/.

[2] Gratis online enclopædi. http://www.wikipedia.org/.

[3] Hejmmesiden for first lego league. www.firstlegoleauge.org.

[4] Hejmmesiden for pixelinespillene. http://www.pixelie.dk.

[5] Hjemmeside for aibo i europa. http://www.aibo-europe.com/.

[6] Hjemmeside for eidos interactive. http://www.eidosinteractive.co.uk/.

[7] Hjemmeside for firmaet Friendly Robotics. http://www.friendlyrobotics.com.

[8] Hjemmeside for k-team der har udviklet khepera ii. www.k-team.com.

[9] Hjemmesiden for gamespot en side hvor mange spil er anmeldt. http://www.gamespot.com/.

[10] Hjemmesiden for konferencerne nordichi. www.nordichi.org.

[11] Hjemmesiden for robocup. http://www.robocup.org.

[12] Legos hjemmeside. http://www.lego.com.

[13] Lidt om spybots. http://www.lugnet.com/~414/spybotics_internals.

[14] Om not quite c til spybotics. http://bricxcc.sourceforge.net/nqc/ eller http://www.baumfamily.org/nqc_old/index.html.

[15] Støvsugeren trilobites underside hos elextrolux. http://trilobite.electrolux.dk.

[16] R. Azuma. A survey of augmented reality, 1995.

[17] Mark Billinghurst, Hirkazu Kato, and Ivan Poupyrev. The magicbook - moving seamlessly between reality and virtuality. *IEEE Comput. Graph. Appl.*, 21(3):6–8, 2001.

[18] Sven Bruel and Niels Åge Nielsen. *Gyldendals Fremmedordbog*. Gyldendalske Boghandel, Nordisk Forlag A.S., Copenhagen, 10 edition, 1989.

[19] Ole Caprani. Natur/teknik forståelse - en sproglig udfordring. Artiklen kan findes på http://legolab.daimi.au.dk/Steno.dir/Steno.html.

[20] Ole Caprani. *RCX Manual*, Århus, Denmark, 2003.

[21] Ole Caprani, Jakob Fredslund, Jens Jacobsen, Line Kramhøft, Rasmus B. Lunding, Jørgen Møller Ilsøe, and Mads Wahlberg. *Evolution of Computer Bugs - an Interdisciplinary Team Work*, chapter 10. Springer Verlag, 2002.

[22] Vinton Gray Cerf. Parry encounters the doctor. *DATAMATION*, pages 62–64, July 1973. Conversation between a simulated paranoid and a simulated psychiatist.

[23] S. Ficici, R. Watson, and J. Pollack. Embodied evolution: A response to challenges in evolutionary robotics, 1999.

[24] Jakob Fredslund. Intelligent adfærd uden intelligens. Master's thesis, Aarhus Universitet, februar 1998.

[25] Jakob Fredslund and Maja Mataric. Robot formations using only local sensing and control. In *Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation (IEEE CIRA 2001)*, pages 308–313, 2001.

[26] Tim Kindberg George Coulouris, Jean Dollimore. *Distributed Systems: Concepts and Design, 3rd edition*. Addison Wesley, 2000.

[27] Brian P. Gerkey and Maja J Mataric. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, special issue on Advances in Multi-Robot Systems, 18(2):758–786, oct 2002.

[28] Eugene Hecht. *OPTICS*. Addison Wesley Longman Inc., third edition, 1991.

[29] Hitachi. *Hitachi H8/300 programming manual*, ????

[30] David W. Hogg, Fred Martin, and Mitchel Resnick. Braitenberg creatures. Technical Report TR, MIT Media Labs, 1991, 10 pages, 1991.

[31] John H. Holland. *Emergence: from chaos to order*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[32] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. *Lecture Notes in Computer Science*, 929:704–??, 1995.

[33] Nick Jakobi. Evolutionary robotics and the radical envelope of noise hypothesis. *Adaptive Behavior*, 6:325–368, 1997.

[34] K-team. Khepera user manual version 5.02, March 1999.

[35] Steven L. Kent. *The Ultimate History of Video Games*. Parma Publishing, 2001.

[36] Hanne A Jeppesen Lene B Christensen og Lars Henningsen Klaus Testrup, Ole Caprani. Robotterne går sig en tur, 2002. Undervisningshæfte udgivet i forbindelse med video.

[37] Lars Konzack. *Softwaregenrer*. Aarhus Universitetsforlag, 1999.

[38] Thiemo Krink. Chapter 1. introduction to evolutionary algorithms. Draft. Udleveret i forbindelse med kurset *Topics in Evolutionary Computation*.

[39] Thiemo Krink. Motivation networks - a biological model for autonomous agent control. *Journal of Theoretical Biology*, 2000.

[40] Walker H. Land, Jr., Margaret Bryden, Daniel W. McKee, Joseph Y. Lo, and Frances R. Anderson. Performance tradeoff between evolutionary computation (ec)/adaptive boosting (ab) hybrid and support vector machine breast cancer classification paradigms. In David B. Fogel, Mohamed A. El-Sharkawi, Xin Yao, Garry Greenwood, Hitoshi Iba, Paul Marrow, and Mark Shackleton, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 187–192. IEEE Press, 2002.

[41] Ian W. Marshall Lionel Sacks, Antonio E. Gonzalez-Velazquez. Simple spontaneous mechanism for flexible data communication in wireless ad hoc sensor networks. *???*, 2002.

[42] Jiming Liu. *Automomous Agents and Multi-agent systems*. World Scientific, 2001.

[43] Wendy E. Mackay. Augmented reality: linking real and virtual worlds: a new paradigm for interacting with computers. In *Proceedings of the working conference on Advanced visual interfaces*, pages 13–21. ACM Press, 1998.

[44] P. Maes. Modeling adaptive autonomous agents. *Artificial Life, I*, (1&2)(9), 1994.

[45] F. Martin. A toolkit for learning: Technology of the mit lego robot design competition, 1994.

[46] M. Mataric. Behavior-based control: Examples from navigation, 1997.

[47] Maja Mataric and Dave Cliff. Challenges in evolving controllers for physical robots. Technical Report CS-95-184, 1995.

[48] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*, chapter 7,8,10,12. Springer, Berlin, 2000.

[49] A. Nareyek. Intelligent agents for computer games. In *Proceedings of the Second International Conference on Computers and Games (CG 2000)*, 2000.

[50] R. Pfeifer. Teaching powerful ideas with autonomous mobile robots, 1997.

[51] Craig Reynolds. Steering behaviors for autonomous characters, 1999.

[52] C.W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Siggraph87*, 1987.

[53] Richard Rouse and Steve Ogden. *Game Design Theory and Practice*. Wordware Publishing Inc., 2000.

[54] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice-Hall, Inc., 2003. Secont Editon.

[55] Simon Egenfeldt-Nielsen & Jonas H. Smith. *Den Digitale Leg - om børn og computerspil*. Hans Reitzels Forlag, 2000.

[56] Robert St. Amant and R. Michael Young. Links: artificial intelligence and interactive entertainment. *intelligence*, 12(2):17–19, 2001.

[57] William Stallings. *Data and computer communications (6th ed.)*. Prentice-Hall, Inc., 2000.

[58] K. Støy. Using situated communication in distributed autonomous mobile robots. In *Proceedings, 7th Scandinavian Conf. on Artificial Intelligence*, pages 44–52, Odense, Denmark, 2001.

[59] Bjørn Nielsen Sune Kristensen, Troels Andersen and Kaj Grønbæk. Battleboard 3d - an augmented reality boardgame using lego for the physical and digital pieces. 2003.

[60] Andrew S. Tanenbaum. *Structured Computer Organization (4rd ed.)*. Prentice-Hall, Inc., 1999.

[61] LEGO Product Technology. *LEGO Spybotics ROM docomentation*, 2002.

[62] Klaus Testrup. Lystfiskere i mediehavet, januar 2004. Kompendium, version 8 Jydsk Pædagog-Seminarium Randers.

[63] Richard A. Watson, Sevan G. Ficici, and Jordan B. Pollack. Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 335–342, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press.

[64] A. Winfield. Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots, 2000.

[65] Michael Woolridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons Ltd., 2002.