

RoboCup – tretí rozmer
(Dokumentácia k projektu)

Tím č. 17: Neurotics

Členovia tímu: Aleš Katona, Gabriel Braniša, Peter Čimo,
Tomáš Labuda, Juraj Šimon, Ján Kolesár

Študijný program: IS/SI

Ročník prvý, inžinierske štúdium

Email: team17.robocup@gmail.com

Web: <http://www2.dcs.elf.stuba.sk/TeamProject/2007/team17is-si/>

Vedúci projektu: Ing. Marián Lekavý

Ak. rok: 2007/2008

Obsah

Obsah	i
Zoznam obrázkov	iii
1. Úvod.....	1
1.1 Účel dokumentu	1
1.2 Cieľ projektu	1
1.3 Zadanie	1
1.4 RoboCup.....	2
2. Analýza.....	3
2.1 Pravidlá RoboCup-u.....	3
2.1.1 Kategória Soccerbot.....	3
2.1.2 Kategória 3D development	4
2.2 Simulačné prostredie	5
2.2.1 Štruktúra robota.....	6
2.2.2 Štruktúra prostredia robota.....	8
2.2.3 Komunikačný protokol	9
2.2.4 Zhodnotenie.....	13
2.3 Analýza rôznych hráčov a tímov.....	13
2.3.1 Sama3D [4]	13
2.3.2 SEU-3D [5]	14
2.3.3 UI-AI [10]	17
2.3.4 Agent Zigorat [6].....	19
2.3.5 DNU Explorer [7].....	21
4.1.1 Fantasia [8].....	21
4.1.2 Naito Striker [9].....	22
4.1.3 Agent Hazard.....	23
4.2 Zhodnotenie analýzy	26
5. Špecifikácia požiadaviek.....	27
6. Hrubý návrh.....	28
6.1 Štruktúra agenta	28
6.2 Model vstávania	29
6.3 Využitie evolučných algoritmov	29
6.3.1 Reprezentácia jedinca	30
6.3.2 Fitness	31
6.3.3 Spôsob výberu rodičov	31
6.3.4 Operácie kríženia a mutácie	31
6.3.5 Výber jedincov do novej generácie	32
7. Podrobný návrh.....	33
7.1 Návrh evolučného algoritmu.....	33
7.1.1 Reprezentácia jedinca	33
7.1.2 Fitness	34
7.1.3 Spôsob výberu rodičov	34
7.1.4 Operácie kríženia a mutácie	34
7.1.5 Výber jedincov do novej generácie	35
7.1.6 Vykonanie jedinca.....	35
8. Prototyp	36
8.1 Inštalácia	36
8.2 Štruktúra hráča	36
8.3 Činnosť hráča	37
8.4 Popis behaviorov	37

8.5	Záver.....	39
8.5.1	Overenie architektonického návrhu agenta.....	39
8.5.2	Overenie implementácie agenta <i>Zigorat</i>	39
8.5.3	Overenie nefunkcionálnych požiadaviek na server	39
9.	Revízia č.1	40
9.1	Kapitola 2.2 Simulačné prostredie	40
9.1.1	Soccer Server.....	40
9.1.2	Monitor	41
9.2	Kapitola 5. Špecifikácia požiadaviek	41
9.2.1	Schopnosti robota	41
9.2.2	Komunikácia robota.....	41
9.2.3	Architektúra robota.....	41
9.2.4	Špecifikácia evolúcie	42
10.	Literatúra	43
11.	Prílohy	45
11.1	Príloha A - Príklad dát prijatých zo servera	1
11.2	Príloha B - Inštalácia servera	1
11.2.1	Povolenie <i>Universe</i> a <i>Multiverse</i> repozitárov	1
11.2.2	Inštalácia potrebných balíčkov	1
11.2.3	Vytvorenie liniek pre ruby.....	1
11.2.4	Stiahnutie zdrojových súborov servera z CVS repozitára	1
11.2.5	Build a inštalácia servera	1
11.2.6	Inštalácia modelov a textúr	2
11.2.7	Spustenie servera	2
11.2.8	Spustenie defaultného agenta.....	2
11.3	Príloha C - Popis tried evolúcie	1
11.4	Príloha D – CD nosič.....	1

Zoznam obrázkov

Obr. 1	Architektúra simulátora [14].....	5
Obr. 2.	Štruktúra robota “soccerbot056”	6
Obr. 3	Pántový kĺb (Hinge joint)	6
Obr. 4	Univerzálny kĺb (Universal joint)	7
Obr. 5	Orientácia osí jednotlivých kĺbov robota [16]	8
Obr. 6	Štruktúra ihriska.....	8
Obr. 7	Znázornenie uhlov poskytovaných perceptorom pre polohu červenej bodky	10
Obr. 8	Orientácia súradnicových osí robota.....	11
Obr. 9	Extrahovanie ZMP parametrov pre určenie výslednej pozície	14
Obr. 10	Učenie neurálnej siete.....	14
Obr. 11	Určovanie parametrov za pomoci neurónovej siete	14
Obr. 12	Vnútorne usporiadanie agenta.....	15
Obr. 13	Model ovládača chôdze	16
Obr. 14	Kontinuálny pohyb robota	16
Obr. 15	štruktúra seu-3d-toolkit.....	17
Obr. 16	Komunikačná vrstva.....	17
Obr. 17	Model hráča soccerbot056.....	19
Obr. 18	Poprepájanie zdrojového kódu na úrovni súborov.....	20
Obr. 19	Schéma agenta DNU Explorer.....	21
Obr. 20	Schéma agenta Naito Striker.....	22
Obr. 21	Návrh hierarchie modulov správania	23
Obr. 22	Information Storage.....	25
Obr. 23	Vrstvy modelov udalostí.....	29
Obr. 24	Príklad koreňového stromu. Funkcia $F(x)=x*(1+x)$	30
Obr. 25	Príklad kríženia dvoch stromov. Zátvorky označujú náhodne zvolené body kríženia.	31
Obr. 26	Koreňový strom s Readovým lineárnym kódom každého podstromu (čísla mimo stromu)	33
Obr. 27	Kríženie dvoch m-chromozómov.....	34
Obr. 28	Kríženie dvoch koreňových stromov	34
Obr. 29	UML diagram štruktúry behaviorov	37

1. Úvod

1.1 Účel dokumentu

Tento dokument bol vytvorený v rámci predmetu Tímový projekt v akademickom roku 2007-2008. Venuje sa virtuálnej simulácii 3D robotického futbalu pod názvom RoboCup 3D. Dokument je členený na jednotlivé kapitoly nasledovne. V prvej kapitole si určujeme cieľ nášho projektu, opisujeme zadanie a ponúkame stručný úvod do prostredia RoboCup-u. Druhou kapitolou je analýza. V nej sa najprv pozrieme na kategórie do ktorých sa delí simulačná liga RoboCup-u. Ďalej si povieme niečo o pravidlách a samotnom simulačnom prostredí. Nakoniec sa pozrieme na niekoľko tímov resp. hráčov robotického futbalu a podrobnejšie si ich opíšeme. Tretia kapitola predstavuje našu špecifikáciu požiadaviek. V štvrtej kapitole opisujeme náš hrubý návrh humanoidného hráča. Využijeme pri ňom znalosti z analýzy z kapitoly tri a prispievame svojou vlastnou ideou ako vylepšiť pohyby humanoida na základe evolučného učenia sa. Piata kapitola obsahuje podrobný návrh predovšetkým evolučných algoritmov, v šiestej je opísaný náš prototyp.

1.2 Cieľ projektu

Cieľom projektu je vytvoriť humanoidného hráča so základnými orientačnými a pohybovými schopnosťami (chôdza, státie, postavenie sa). Ako ideologický základ je vhodné využiť prácu a poznatky minuloročného tímu. Nový humanoidný hráč bude pracovať na novej verzii serveru vo verzii 0.5.6. Je potrebná vyhľadať iných humanoidných hráčov, jedného z nich si vybrať a na jeho základe zostaviť hráča vlastného. Implementovať mu základné schopnosti a poskytnúť tak určitý základ pre prácu tímov v budúcich rokoch. Hráč má byť modulárny a schopný rozšírenia v budúcich rokoch.

1.3 Zadanie

Téme RoboCup, presnejšie lige simulovaného robotického futbalu sa naši študenti venujú už osem rokov. Tímy študentov, či už v rámci umelej inteligencie alebo tímového projektu, sa snažia vytvárať a vylepšovať programy, ktoré simulujú správanie sa futbalového hráča. Každý tím sa v rámci obmedzení, určených pravidlami hry futbal a špecifikami simulačného prostredia, snaží vytvoriť čo najlepšieho hráča. Mužstvo, vytvorené z takýchto hráčov, by malo vyhrať nad mužstvom súpera. O súťaži a doterajšej činnosti je dost' popísané aj na stránke STU [11].

Simulácia futbalu pôvodne prebiehala iba v dvoch rozmeroch. Pre zvýšenie reálnosti simulácie bolo vytvorené 3D simulačné prostredie, ktoré rozširuje možnosti hry. 3D simulačné prostredie sa pomerne výrazne líši od doposiaľ používaného 2D prostredia, a to jednak spôsobom simulácie, ale hlavne možnosťami ktoré poskytuje hráčom.

Hlavným cieľom projektu bude vytvoriť hráča pre 3D simuláciu, ktorý dokáže plnohodnotne využívať možnosti poskytované simulačným prostredím. Úlohou teda bude prevziať jednoduchého hráča vytvoreného na našej fakulte v minulom roku a doplniť do neho komplexnejšie typy správania a rozhodovania. Keďže sa predpokladá ďalšie rozširovanie hráča v ďalších rokoch, dôležitými požiadavkami sú prehľadnosť a ďalšia rozširovateľnosť, a to na úrovni návrhu aj implementácie. Dôraz pri vytváraní hráča by mal byť kladený najmä na dobre prepracované a

odladené nižšie schopnosti hráča, ktoré umožnia hráčovi spracovávať vnemy z prostredia a efektívne konať v prostredí (pohybovať sa, pracovať s loptou). Pri návrhu a implementácii bude tiež možné (a žiadané) čerpať z veľkého množstva prístupov existujúcich v 2D simulácii.

Zimný semester je vyhradený na oboznámenie sa s celým simulačným prostredím a hráčom ktorý sa bude rozširovať, a takisto s existujúcimi hráčmi (2D aj 3D), ďalej návrhu a prototypovej realizácii hráča. Dôležitou súčasťou bude vytvorenie plánu implementácie a overovania prístupu v nasledovnom semestri. V letnom semestri nás čaká dokončenie realizácie návrhu a jeho overovanie. Nemenej podstatnou časťou projektu bude vytvorenie dokumentácie, ktorá poskytne tímom v ďalších rokoch odrazový mostík pri použití vytvoreného hráča.

1.4 RoboCup

RoboCup je medzinárodná súťaž autonómnych robotov softvérových agentov. Súťaž je určená na podporu výskumu a výučby umelej inteligencie, robotiky a ďalších príbuzných oblastí. *Robot World Cup Soccer Games and Conferences*, ide o neziskovú organizáciu, ktorá každoročne usporadúva turnaje v robotickom futbale. Snahou tejto spoločnosti je podporovať vývoj rôznych technologických oblastí pomocou riešenia štandardnou metódou, pri ktorej je možné využiť obrovské množstvo technológií. Na tento účel programátori zvolili futbal. Základnou ideou je vytvoriť čo najviac dokonalých fyzických, ale i syntetických, čiže programovo realizovaných agentov, ktorí dokážu hrať futbal na vysokej úrovni. Míľnikom, o ktorý RoboCup-u ide, je zostrojenie tímu humanoidných robotov, ktorí by hrali proti najlepšiemu "ľudskému" tímu podľa oficiálnych pravidiel FIFA. Predpovede organizácie RoboCup hovoria o uskutočnení tohto zápasu okolo polovice tohto storočia.

RoboCup Federation je medzinárodná nezisková organizácia zaregistrovaná v Ženeve za účelom podpory, propagácie a posilňovania výskumu RoboCup-u. Túto iniciatívu v súčasnosti nasleduje približne 1500 výskumníkov v 17 rôznych krajinách. Do súťaže sa môže zapojiť každý tím (resp. jednotliviec), bez ohľadu na to, či je alebo nie je zameraný na samotný výskum. Niektoré tímy sa zúčastňujú súťaže iba zo zábavy (najmä v simulačnej lige), iné sú zamerané na seriózný výskum, pričom svoje výsledky pravidelne publikujú.

Existuje liga simulovaného futbalu pre 2D, kde sú agenti v tvare kruhu v počte jedenásť na hracej ploche. V 3D existujú dva typy simulovaného futbalu a to reprezentácia agenta guľičkou, čo je pokračovanie, nadstavba 2D simulovaného futbalu. Tento typ postupne zaniká a nahradila ho simulácie 3D humanoidov v počte troch hráčov na jeden tím. Simulačné prostredie disponuje realistickým simulovaním fyzikálnych zákonov pomocou ODE knižnice.

2. Analýza

V nasledujúcej kapitole si postupne predstavíme najprv základné kategórie súťaže RoboCup a ich pravidlá. Ďalej sa oboznámime so samotným simulačným prostredím a modelom nového humanoidného robota. Následne si predstavíme niekoľko tímov pôsobiacich v oblasti simulovaného robotického futbalu.

2.1 Pravidlá RoboCup-u

Bohužiaľ na oficiálnych stránkach RoboCup-u [1] sme nenašli žiadne užitočné a aktuálne informácie. Kategória 3D simulovaného futbalu sa od roku 2005 kedy boli tieto informácie naposledy aktualizované, dosť zmenila (predovšetkým zmena modelu hráča). Na stránke ligy sa však nachádzajú aktuálnejšie informácie, konkrétne pravidlá z turnaju RoboCup 2007 v Atlante [3]. V Atlante sa použil server rcserver3d-0.5.6 spustený pod operačným systémom SuSE Linux 10.1. Server a monitor bežali na samostatných počítačoch, a každý tím mal k dispozícii jeden počítač pre svojich hráčov.

2.1.1 Kategória Soccerbot

Jedná sa o kategóriu v ktorej proti sebe súťažia dva tímy hráčov.

Tými môžu použiť jedného z dvoch typov hráča:

1. *soccerbot055.rsg* – Je to verzia bota, ktorá prišla so serverom rcserver3d-0.5.5.
 - jednodielny trup
 - žiadne obmedzenia kĺbov – môžu sa otáčať aj do neprirodzených polôh
 - TCH senzor – poskytuje len informáciu o tom že sa niečo niečoho dotýka
 - nemá detekciu kolízií rúk
2. *soccerbot056.rsg* – Tento bot je súčasťou servera rcserver3d-0.5.6.
 - dvojdielny trup
 - obmedzenia kĺbov – nemôžu sa otáčať do neprirodzených polôh
 - FRP senzor – poskytuje informáciu o sile, ktorá pôsobí pri dotyku a o mieste jej pôsobenia.
 - detekcia kolízií rúk

Pozn.: Obidvaja boti majú rovnakú hmotnosť.

Štruktúra zápasov je nasledovná:

- Jeden tím je zložený z 3 hráčov (v pravidlách nie je napísané či je presne určený brankár, ale pravdepodobne to bude hráč s číslom jeden).
- Hra je rozdelená na dva polčasy. Každý trvá 7,5 min.
- Počas súťaže tímy môžu zmeniť alebo nahradiť svoje zdrojové kódy. Tieto zmeny budú však vykonané na vlastné riziko. Zmena je možná len medzi jednotlivými zápasmi, nie počas hry.
- Ak hráč nebude fungovať organizátori ho nebudú rozbehávať. (Tímy totiž nemusia prísť na súťaž osobne, ale môžu svojich hráčov poslať.)

Na hru dohliada ľudský rozhodca, pričom niektoré fauly sú automaticky detekované simulátorom, no ten nemôže detekovať všetky. Rozhodca má právo udeliť voľný kop poškodenému tímu. Za faul sa pokladá napríklad (no nie len):

- Ak jeden z hráčov úmyselne chytí, alebo sa dotkne lopty rukou, okrem brankára.
- Ak niektorý hráč leží na lopte viac ako 10 sekúnd.
- Ak hráč úmyselne udrie alebo postrčí súperovho hráča.
- Ak jeden tím obkľúči loptu tak, že druhý tím nemá na loptu dosah.
- Ak hráči blokujú bránku tak že sa lopta nemôže dostať dnu (stena z hráčov v bránke).
- Ak tím úmyselne blokuje pohyb hráčov.
- Čokoľvek, čo sa javí ako porušenie fair play, môže byť tiež považované za faul. Záleží na posúdení rozhodcu.

Cieľom hry je hrať futbal podľa obvyklého vnímania fér hry a obmedzení daných virtuálnym simulovaným svetom 3D simulátora. Obchádzanie týchto pravidiel je považované za porušenie povinnosti hrať fér hru a počas turnajových zápasov je to prísne zakázané.

Za porušenie fair play sa považuje napríklad:

- Používanie hráčov iného tímu.
- Zhlcovanie servera posielaním neúmerneho počtu správ od klienta.
- Priama komunikácia hráčov použitím iných komunikačných spôsobov, ako napríklad medzi procesová komunikácia.
- Práca so súťažnými počítačmi alebo ich úmyselné reštartovanie.

Čokoľvek z uvedeného je prísne zakázané. Iné stratégie môžu byť označené za porušenie fair play, po konzultácií s komisiou zaoberajúcou sa pravidlami. Obzvlášť deštruktívne narušenie operácií súperových agentov, alebo získanie výhody spôsobom iným, ako explicitne ponúkaným simulátorom, je považované za porušenie fair play. Ak si účastníci súťaže nie sú istý ohľadne použitia niektorej metódy, konzultujú to s komisiou zaoberajúcou sa pravidlami pred začiatkom turnaja. Ak sa počas turnaja zistí, že niektorý tím používa ne fér praktiky, bude tento tím okamžite vylúčený z turnaja.

V pravidlách sa nepíše nič o koučovi alebo trénerovi. Predpokladáme teda že ešte nie je implementovaný v servery a teda ani povolený (použiteľný).

2.1.2 Kategória 3D development

Kategória 3D development je zameraná na urýchlenie vývoja nízko úrovňových schopností hráča, aby tými mohli čo najrýchlejšie prejsť na vývoj vyšších schopností hráčov. Presné pravidlá ešte nie sú definované v použiteľnom rozsahu, ale na oficiálnej stránke ligy [2] sú spomenuté niektoré súťaže, z tejto kategórie:

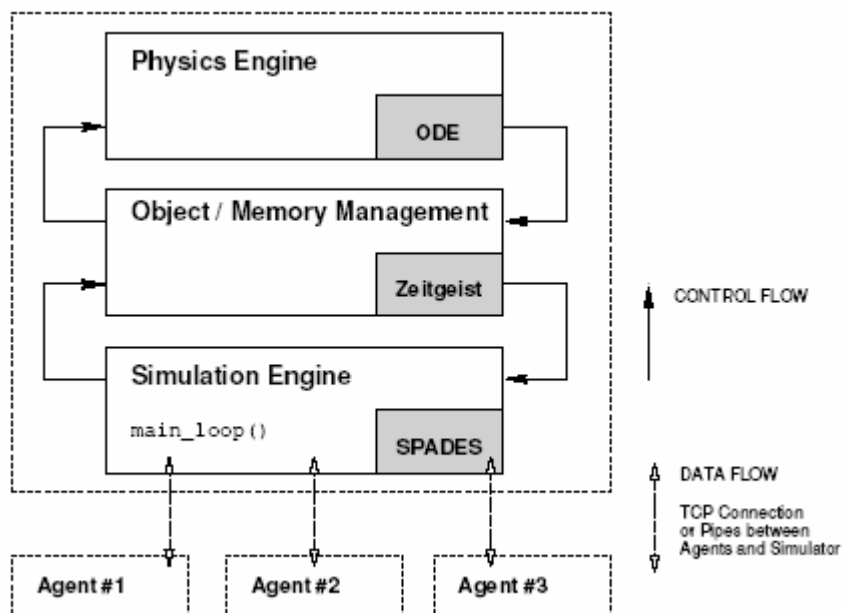
Technické súťaže - môžu obsahovať kráčanie, kopanie, vstávanie, penalty, ...

Futbalové zápasy 5-vs-5 - Keďže zápas 5-vs-5 je komplexná úloha, boli pravidlá pre rok 2007 upravené. Hráči mohli narábať s loptou akoukoľvek časťou tela (vrátane rúk). Tiež sa mohli pohybovať kráčaním, behaním, plazením, ...

2.2 Simulačné prostredie

Ako simulačné prostredie sa využíva SPARK [14], multi-agentový simulačný systém hmotných agentov v troj-dimenzionálnom prostredí. Nasledujúci obrázok (obr. 1.) znázorňuje hlavné časti simulačného systému:

- *Prostriedok pre fyzikálnu simuláciu (Physics Engine)* – implementovaný pomocou knižnice ODE. ODE slúži na fyzikálnu simuláciu komplexných objektov zložených z pevných častí prepojených kĺbmi.
- *Prostriedok na správu objektov (Object / Memory Management)* – implementovaný pomocou frameworku Zeitgeist. Prostriedok umožňuje flexibilný a jednotný prístup k jednotlivým komponentom simulátora:
 - objektom reprezentujúcim simulovanú scénu (sprístupnené ODE objekty),
 - objektom zabezpečujúcim základnú funkcionálnu simulátora.
- *Simulačný prostriedok (Simulation Engine)* – zabezpečuje hlavný cyklus vykonávania a interakciu medzi agentmi a simulátorom. Simulácie väčšieho počtu agentov rozmiestnených na viacerých počítačoch sú ovplyvňované faktormi ako aktuálne zaťaženie siete, rôzne typy latencií a rôzne rýchlosti CPU. Simulácie sú tak často nereprodukovateľné. Z tohto hľadiska je prostriedok implementovaný dvoma spôsobmi:
 - Implementácia pomocou SPADES – berie do úvahy vyššie spomenuté faktory a umožňuje reprodukovateľné simulovanie.
 - Jednoduchá implementácia – vyššie uvedené faktory jednoducho ignoruje, nezaručuje reprodukovateľné simulovanie, ale poskytuje vyšší výkon.

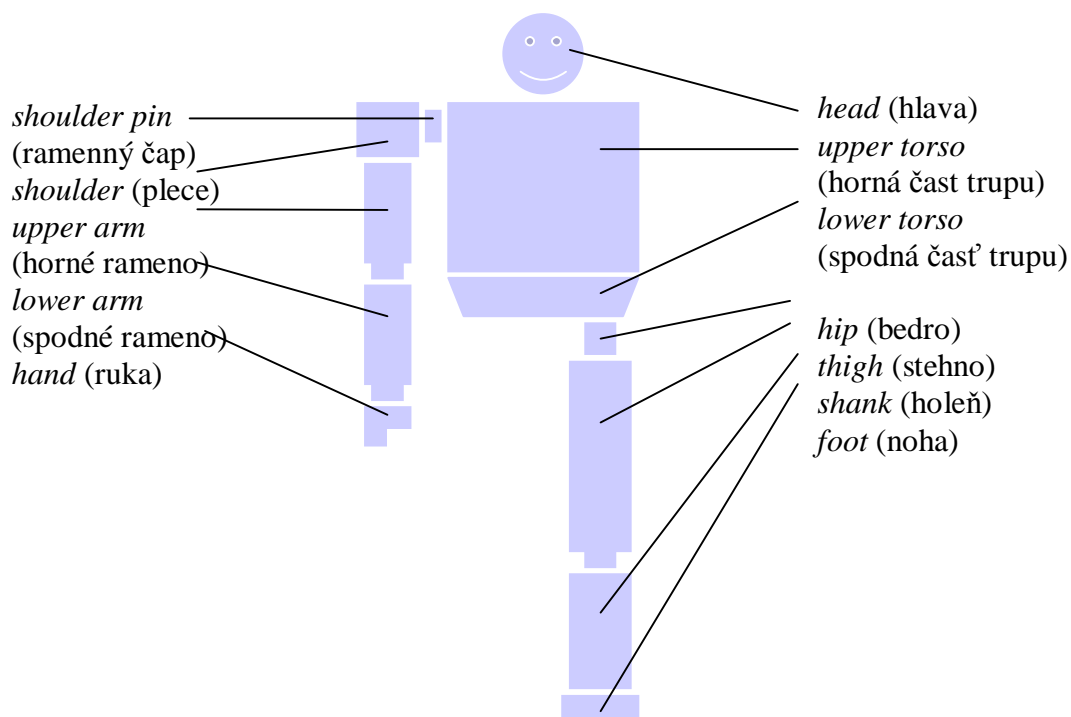


Obr. 1 Architektúra simulátora [14]

Aktuálny server zabezpečujúci simuláciu robotického futbalu je implementovaný práve systémom SPARK bez použitia SPADES. V súčasnosti je dostupný server vo verzii 0.5.6 a ďalší popis bude preto aktuálny pre túto verziu.

2.2.1 Štruktúra robota

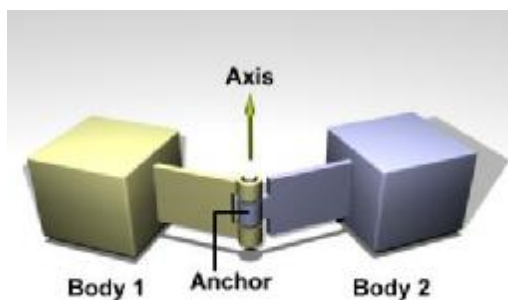
Ako už bolo spomenuté v predchádzajúcej kapitole, simulačný prostriedok SPARK je univerzálny a umožňuje simuláciu rôznych typov hráčov (s rôznymi tvarmi, kĺbmi, efektormi, perceptorami). V súčasnosti sa používa typ robota “soccerbot056” a práve jeho štruktúra bude ďalej popísaná. Nasledujúci obrázok (obr. 2.) znázorňuje jednotlivé časti robota.



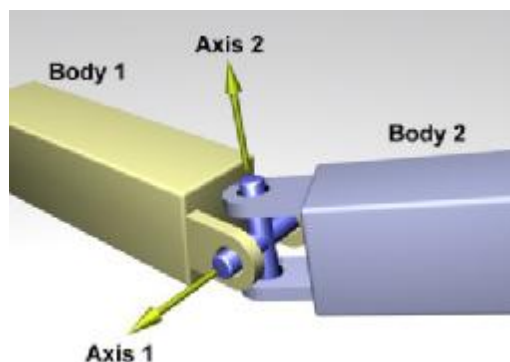
Obr. 2. Štruktúra robota “soccerbot056”

Robot je pre potreby pohybu vybavený niekoľkými kĺbmi. V popisovanom type robota sa využívajú dva typy pohyblivých kĺbov definovaných knižnicou ODE [15]:

- *Pántový kĺb* (*Hinge joint, HJ*) – poskytuje jeden stupeň voľnosti otáčania (obr. 3.),
- *Univerzálny kĺb* (*Universal joint, UJ*) – poskytuje dva stupne voľnosti otáčania (obr. 4.).



Obr. 3 Pántový kĺb (Hinge joint)



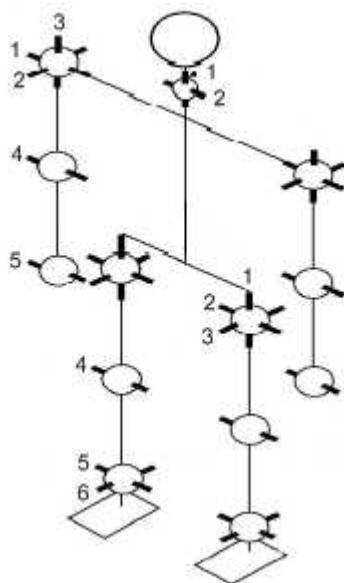
Obr. 4 Univerzálny kĺb (Universal joint)

Kĺby robota sú vybavené perceptorami, ktoré poskytujú informácie o stave kĺbov a efektorami, pomocou ktorých je možné stav kĺbov meniť. Prepojenie jednotlivých častí robota kĺbmi, názvy perceptorov a efektorov jednotlivých kĺbov popisuje nasledujúca tabuľka (tab. 1.).

prepojené časti tela robota		typ kĺbu	perceptor*	efektor*
head	upper torso	FJ		
lower torso	upper torso	HJ		
shoulder	upper torso	UJ	laj1_2	lae1_2
upper arm	shoulder	HJ	laj3	lae3
lower arm	upper arm	HJ	laj4	lae4
hand	lower arm	FJ		
hip	lower torso	HJ	llj1	lle1
thigh	hip	UJ	llj2_3	lle2_3
shank	thigh	HJ	llj4	lle4
foot	shank	UJ	llj5_6	lle5_6
<i>FJ – Fixed joint, nepohyblivý kĺb</i> <i>HJ – Hinge joint</i> <i>UJ – Universal joint</i> * – názvy perceptorov a efektorov sú uvedené uvažujúc ľavé končatiny (pre pravé by sa názvy začínali písmenom “r”)				

Tab. 1. Prepojenie častí tela kĺbmi

Orientácia osí jednotlivých kĺbov je zrejmá z nasledujúceho obrázku (obr. 5.) (pohyblivé kĺby na krku a rukách sa v súčasnej verzii robota nevyskytujú).

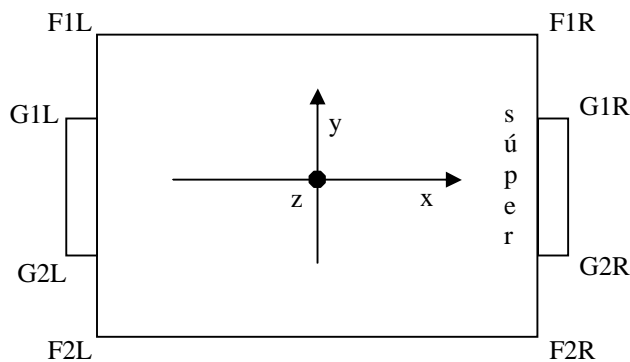


Obr. 5 Orientácia osí jednotlivých kĺbov robota [16]

2.2.2 Štruktúra prostredia robota

Prostredie robota pri simulácii pozostáva z:

- spoluhráčov,
- protivráčov,
- lopty,
- ihriska (obr. 6.):
 - 4 rohové zástavky ($F1L$, $F2L$, $F1R$, $F2R$),
 - 4 okraje bránok ($G1L$, $G2L$, $G1R$, $G2R$).



Obr. 6 Štruktúra ihriska

2.2.3 Komunikačný protokol

Každý hráč (robot) reprezentuje v architektúre klient-server jedného klienta. Komunikácia so serverom prebieha prostredníctvom protokolov TCP a UDP (prednastavený je TCP protokol). Správy vymieňané medzi klientom a serverom majú tvar tzv. S-výrazu. Ide o konvenciu na reprezentáciu semištruktúrovaných dát¹. Nasleduje popis komunikačného protokolu medzi klientom a serverom [17][18].

Server->klient

Klient dostáva od servera informácie z perceptorov robota. Správy sú posielané v diskretných časových intervaloch 0.02s. Nasleduje popis správ prijímaných z jednotlivých perceptorov z hľadiska syntaxe a sémantiky.

Príklad kompletnej správy prijatej klientom sa nachádza v prílohe A.

2.2.3.1 TimePerceptor

Perceptor poskytuje informácie o aktuálnom simulačnom čase (čas od začiatku simulácie, nie hry).

Vzor: (time (now <simulačný_čas>))

Príklad: (príloha A)

2.2.3.2 GameState perceptor

Perceptor poskytuje informácie o aktuálnom stave hry (hrací čas a mód hry).

Vzor: (GS
(time <hrací_čas>)
(pm <mód_hry>))

- <mód_hry> – môže nadobúdať jednu z hodnôt: *BeforeKickOff*, *KickOff_Left*, *KickOff_Right*, *PlayOn*, *KickIn_Left*, *KickIn_Right*, *corner_kick_left*, *corner_kick_right*, *goal_kick_left*, *goal_kick_right*, *offside_left*, *offside_right*, *GameOver*, *Goal_Left*, *Goal_Right*, *free_kick_left*, *free_kick_right*, *unknown*.

Príklad: (príloha A)

2.2.3.3 Vision preceptor

Perceptor poskytuje informácie o robotom viditeľných objektoch. V súčasnosti má robot 360 stupňové videnie (vidí všetky objekty) a objekty majú podobu hmotných bodov. Pre každý objekt poskytuje perceptor polohu objektu relatívne od robota vo sférických súradniciach. Stredom sférickej sústavy je ťažisko horného trupu robota (upper torso). Pre hráčov okrem polohy poskytuje informáciu o tíme a čísle dresu hráča.

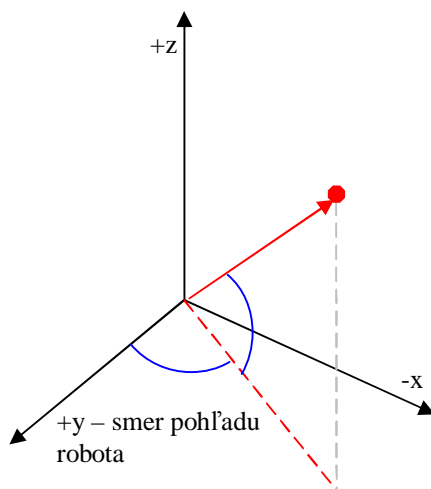
Vzor: (See
(<typ_objektu>
(pol <vzdialenosť> <uhol_1> <uhol_2>))
...
(P

¹Viac v: S-expression, Wikipedia, <http://en.wikipedia.org/wiki/S-expression>

```
(team <názov_tímu>
(id <číslo_hráča>)
(pol <vzdialenosť> <uhol_1> <uhol_2>))
```

- *<typ_objektu>*
 - 4 rohové zástavky (*F1L, F2L, F1R, F2R*) (obr. 6.),
 - 4 okraje bránok (*G1L, G2L, G1R, G2R*) (obr. 6.),
 - 1 lopta (*B*),
 - spoluhráči a súper (P),
- *<vzdialenosť>* – vzdialenosť objektu od ťažiska hornej časti trupu (stred sférickej sústavy),
- *<uhol_1>* – uhol v rovine *xy* v stupňoch (obr. 7),
- *<uhol_2>* – uhol s rovinou *xy* v stupňoch (obr. 7).

Príklad: (príloha A)



Obr. 7 Znázornenie uhlov poskytovaných perceptorom pre polohu červenej bodky

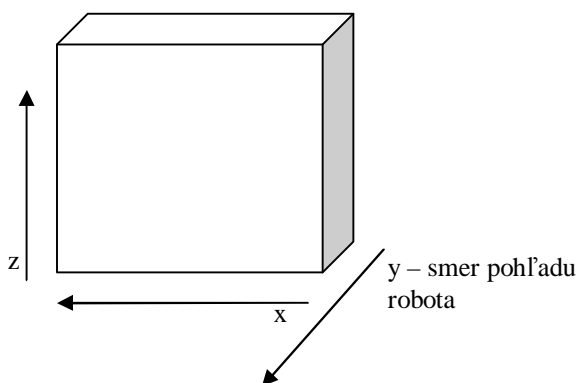
2.2.3.4 GyroRate perceptor

Perceptor je umiestnený v hornom trupe robota (upper torso) a poskytuje informácie o jeho uhlovom zrýchlení okolo jednotlivých osí.

Vzor: (GYR
(name torso)
(rt <x> <y> <z>))

- *<x> <y> <z>* – uhlové zrýchlenia horného trupu robota okolo osí *x*, *y* a *z* v rad.s^{-2} (obr. 8).

Príklad: (príloha A)



Obr. 8 Orientácia súradnicových osí robota

2.2.3.5 Touch perceptor

(Súčasná verzia robota nemá tento perceptor aktívny) Perceptor poskytuje binárnu informáciu, či bola určitá z častí tela účastníkom kolízie.

Vzor: (TCH (name <názov_časti_tela>) (val <hodnota>))

- <hodnota> – 1 v prípade kolízie, inak 0.

Príklad: (TCH (name footleft) (val 1))

2.2.3.6 ForceResistance perceptor

Perceptor poskytuje informáciu o kolíznej sile a priemerný bod pôsobenia kolíznej sily. V prípade, že sa dva povrchy kompletne dotýkajú a existuje množstvo kontaktných bodov, informácia obsahuje len priemerný bod a celkovú silu ako súčet síl pôsobiacu vo všetkých bodoch dotyku.

Vzor: (FRP (n <názov_časti_tela>) (c <px> <py> <pz>) (f <fx> <fy> <fz>))

- <názov_časti_tela> – v súčasnosti poskytuje perceptor informácie z ľavej (*ls*) a pravej (*rf*) nohy,
- <px> <py> <pz> – lokálne súradnice bodu pôsobenia kolíznej sily,
- <fx> <fy> <fz> – kolízny vektor.

Príklad: (príloha A)

2.2.3.7 Joint perceptors (kĺbové perceptory)

Perceptory poskytujú informácie o stave jednotlivých kĺbov robota (aktuálnom otočení podľa osí otáčania).

Hinge joint

Vzor: (HJ
(n <názov>
(ax <uhol>))

- <názov> – názov kĺbového perceptora (tab. 1.),
- <uhol> – uhol otočenia okolo osi otáčania v stupňoch (obr. 5).

Príklad: (príloha A)

2.2.3.8 Universal joint

Vzor: (HJ
 (n <názov>
 (ax <uhol_1>
 (ax <uhol_2>))

- <názov> – názov kĺbového perceptoru (tab. 1.),
- <uhol_x> – uhol otočenia okolo osi x otáčania v stupňoch (obr. 5).

Príklad: (príloha A)

2.2.3.9 Create effector

Efaktor slúži na vytvorenie hráča. Simulačný prostriedok je univerzálny a umožňuje simuláciu rôznych hráčov (s rôznymi tvarmi, kĺbmi, efektormi, perceptorami). Aktuálne sa používa hráč typu "soccerbot056".

Vzor: (scene <cesta_k_definícii_typu_hráča>)

Príklad: (scene rsg/agent/soccerbot056.rsg)

2.2.3.10 Init effector

Efaktor slúži na nastavenie čísla dresu a názvu tímu hráča.

Vzor: (init (unum <číslo_dresu>) (teamname <názov_tímu>))

Príklad: (init (unum 7) (teamname RoboLog))

2.2.3.11 Beam effector

Efaktor slúži na nastavenie pozície hráča na ihrisku pred začatím zápasu.

Vzor: (beam <x> <y> <uhol>)

- < x,y > – pozícia na ihrisku (obr. 6.),
- <uhol> – natočenie hráča v stupňoch (0 stupňov = $x+$, 90 stupňov = $y+$).

Príklad: (beam <10.0> <-10.0> <0.0>)

2.2.3.12 Joint effectors (kĺbové efektory)

Efektory slúžia na obsluhu motorčekov v kĺboch robota. Pre každý kĺb je možné určiť rýchlosť a orientáciu (určuje znamienko rýchlosti) otáčania podľa osí v kĺbe. Zmysel otáčania podľa osí na obr. 5. je pre kladné hodnoty rýchlosti možné určiť pravidlom pravej ruky (palec ukazuje v kladnom smere príslušnej osi (obr. 8.) a zahnuté prsty v smere otáčania).

Hinge joint**Vzor:** (<názov> <rýchlosť_otáčania>)

- <názov> – názov kĺbového efektora (tab. 1.),
- <rýchlosť_otáčania> – rýchlosť otáčania okolo osi otáčania v rad.s^{-1} (obr. 5).

Príklad: (lae3 1.0)**Universal joint****Vzor:** (<názov> <rýchlosť_otáčania_1> <rýchlosť_otáčania_2>)

- <názov> – názov kĺbového efektora (tab. 1.),
- <rýchlosť_otáčania_1> – rýchlosť otáčania okolo osi otáčania 1 v rad.s^{-1} (obr. 5).
- <rýchlosť_otáčania_2> – rýchlosť otáčania okolo osi otáčania 2 v rad.s^{-1} (obr. 5).

Príklad: (lae1_2 1.0 0.0)**2.2.4 Zhodnotenie**

Kapitola poskytla prehľad simulačného prostredia, štruktúry robota a komunikačného protokolu. Vzhľadom na relatívne nedávne uvedenie humanoidného typu robota a príslušného servera, bolo (je) najväčším problémom hľadanie relevantných zdrojov. Do dnešného dňa totiž nie je dostupná kompletná dokumentácia k serveru pre humanoidný typ robota. Ako zdroj informácií pre túto kapitolu bol okrem uvedenej literatúry použitý aj triviálny prototyp na testovanie komunikačného protokolu zasielaním jednoduchých správ. Empirický pôvod majú najmä informácie o funkčnosti *Vision perceptora* a *GyroRate perceptora*. Funkčnosť niektorých perceptorov nebola vôbec overená (*Touch perceptor*, *ForceResistance perceptor*). Jednou z úlohou prototypu bude preto overiť aj ich funkčnosť.

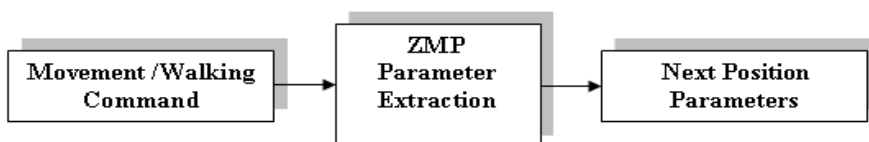
2.3 Analýza rôznych hráčov a tímov

Pred vytvorením našej špecifikácie sme trochu podrobnejšie preštudovali viacero iných tímov a ich hráčov najmä tímov účastniacich sa turnaja v roku 2007 v Atlante.

2.3.1 Sama3D [4]

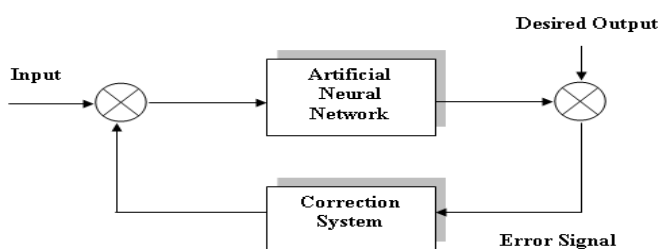
Jedná sa o humanoidného robota ktorý dokáže vykonávať jednoduché pohybové úkony. Na chôdzu využíva neurónovú sieť. Pôvodne robot využíval tzv. ZMP (Zero Moment Point) metódu. Pri ZMP metóde je cieľom udržanie bodu v ktorom je súčet všetkých momentov síl rovné nule, vo vnútri konvexného n-uholníka všetkých dotkových bodov robota s podlahou počas pohybu. Metóda vychádza z matematického modelu kinematiky každej časti schopnej pohybu. Pre každý pohyb je odvodená 3 dimenzionálna rotačná matica a pre každý kĺb je určený translačný vektor. Využitie tejto metódy však vedie k obrovskému zaťaženiu procesora pre každý krok resp. pohyb. Toto zaťaženie autori robota však dokázali znížiť učiacou procedúrou. Na tento účel bola vytvorená neurónová sieť. Sieť bola trénovaná na hodnotách extrahovaných pomocou metódy ZMP. Naučená sieť bola potom schopná určiť ďalší krok (súradnice a ohyby kĺbov) bez potreby zložitých kalkulácií za pomoci ZMP metódy. Robot bol zostrojený a otestovaný v prostredí Webots.

Implementácia robota prebiehala v troch krokoch. V prvom kroku boli určené základné možné pohyby robota v hernom prostredí. Následne boli prepočítané metódou ZMP (v prostredí Matlabu). Z týchto výpočtov boli extrahované údaje pre pohyby kĺbov pre zvolený typ pohybu ako je znázornené na obrázku č. 9.



Obr. 9 Extrahovanie ZMP parametrov pre určenie výslednej pozície

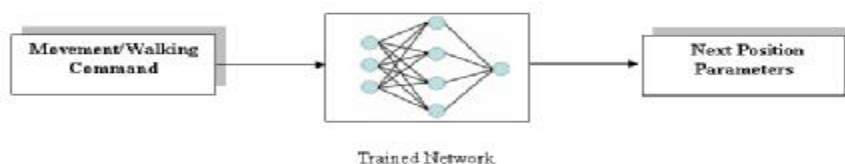
Ďalším krokom bolo vytvorenie a učenie neurónovej siete so vstupmi, ktorými boli dvojice príkaz pre pohyb a k nemu údaje získané ZMP metódou (polohy a rýchlosti kĺbov). Schéma učenia sa je znázornená na obrázku č. 10.



Obr. 10 Učenie neurálnej siete

Sieti je poskytnutý vstup (príkaz pohybu a ZMP údaje). Z výstupu je vypočítaná chyba ktorá je zavedená do korekčného systému. Aplikuje sa korekcia vstupu a sieť sa trénuje ďalej až dokým chyba nie je zanedbateľná.

Po tom čo sa sieť naučí je možné jej priamo vložiť príkaz pohybu a na jej výstupoch dostaneme priamo hodnoty ktoré by poskytla ZMP metóda avšak bez nutnosti toľkých výpočtov, ako ukazuje obrázok.



Obr. 11 Určovanie parametrov za pomoci neurónovej siete

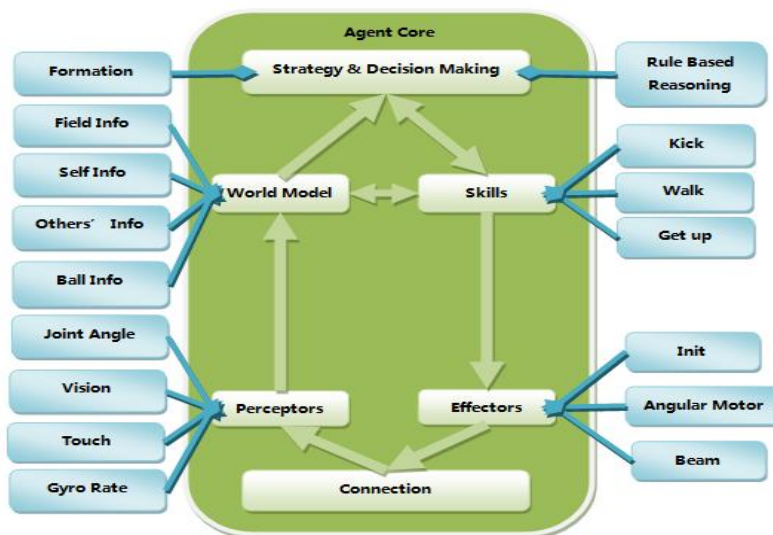
Robot tohto tímu disponoval len základnými pohybovými funkciami na najnižšej úrovni. Robot nebol schopný žiadnych vyšších funkcií ako je práca s loptou či plánovanie a pod.

2.3.2 SEU-3D [5]

Tím vznikol v roku 2005 a úspešne sa zúčastnil viacerých súťaží s robotmi v podobe gule čo bola

staršia verzia serveru pre RoboCup 3d. Po prechode na novšiu verziu serveru a humanoidného hráča sa tím sústredil na vývoj základných schopností, vlastného vývojového prostredia a robustného kódu. V roku 2007 sa umiestnili na prvom mieste na RoboCup Iran Open a na treťom mieste v svetovom robocup3d.

Architektúra agenta je modulárna, formou zásuvných modulov a tzv. singletonov. Teda napríklad pri zmene verzie servera, nie je potrebné meniť celý kód stačí len zmeniť príslušný modul. Na obrázku je znázornená schéma agenta.



Obr. 12 Vnútročné usporiadanie agenta

Zelený box predstavuje jadro robota. Moduly vo vnútri štvorca sú implementované ako singletony pre jednoduchšiu komunikáciu. Moduly mimo štvorca sú implementované ako zásuvné moduly.

World model

Služi na uchovanie si stavu sveta. Z tohto stavu môže robot odvodiť svoju pozíciu P_v a Rotáciu R_v na hracej ploche. Pozícia je odvodzovaná z troch pevných bodov ktoré predstavujú okraje ihriska. Napríklad z horného a dolného ľavého rohu a vrchného pravého rohu p_{1l} , p_{2l} a p_{1r} . Môžeme zostaviť nasledujúce rovnice.

$$\begin{cases} |P_{1l} - P_{1r}| = \rho_{1d} \\ |P_{1l} - P_{2l}| = \rho_{2d} \\ |P_{2l} - P_{1r}| = \rho_{1r} \\ -P_{1x} = -P_{2x} = P_{1rx} = 0.5 \cdot field_length \\ P_{1y} = -P_{2y} = -P_{1ry} = 0.5 \cdot field_width \\ P_{1z} = P_{2z} = P_{3z} = 0 \end{cases} \quad (1)$$

Z ktorých pozícia robota P_v môže byť vypočítaná takto:

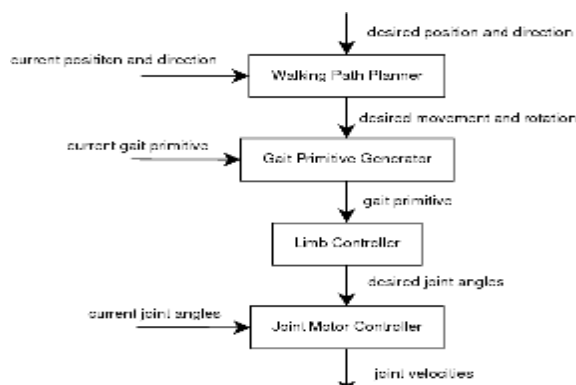
$$P_{1l} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} \frac{\rho_{1d}^2 - \rho_{2d}^2}{2 \cdot field_length} \\ \frac{\rho_{2d}^2 - \rho_{1d}^2}{2 \cdot field_width} \\ \sqrt{\rho_{1d}^2 - (P_x - P_{1lx})^2 - (P_y - P_{1ly})^2} \end{bmatrix} \quad (2)$$

Určenie ťažiska je taktiež dôležité z hľadiska udržania stability robota. Ťažisko môže byť vypočítané vzorcom

$$P_{COM} = \sum P_i \cdot m_i$$

kde P_i je pozícia časti tela a m_i je jeho objem. Ťažisko sa počíta v reálnom čase. Pohľad robota je posunutý na jeho torzo (a nie v hlave).

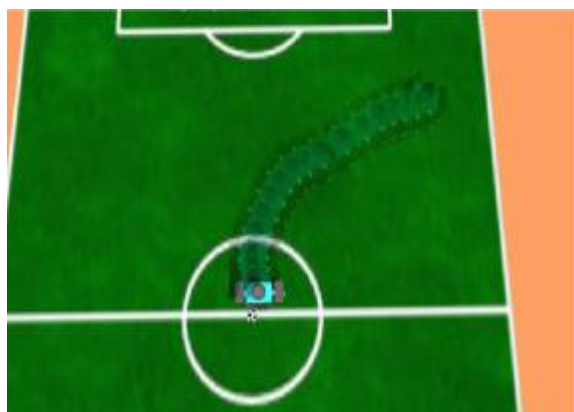
Ovládač vše smerového chodenia agenta má nasledujúcu štruktúru.



Obr. 13 Model ovládača chôdze

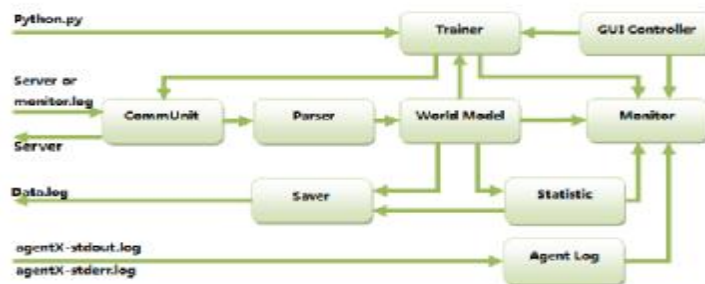
Model ovládača dvojnohej chôdze - Obsahuje viacero vrstiev. *Walking Path Planner* obdrží súradnice a smer chôdze resp. cieľu chôdze. Naplánuje pohyb a odošle ho do *Gait Primitive Generator*. Tento vygeneruje ďalší primitívny pohyb. Tento sa odošle do *Limb Controller* ktorý určí potrebné ohyby kĺbov a potrebné akcie odošle do *Joint Motor Controller* ktorý pohyb kĺbov vykoná.

Výsledkom je hladký neprerušovaný pohyb do všetkých smerov. Robot sa tak nepotrebuje zastaviť keď chce zmeniť smer.



Obr. 14 Kontinuálny pohyb robota

Ako vývojový nástroj bolo použité prostredie *seu-3d-toolkit*. Slúži ako server i nástroj na simuláciu a tréning robotov. Z popisu robota nebolo zjavné či si toto prostredie navrhli sami členovia tímu alebo či sa jedná o určité všeobecné testovacie a vývojové prostredie.



Obr. 15 štruktúra seu-3d-toolkit.

Hlavné výhody tohto prostredia:

- záznam zápasu v priamom aj spätnom smere
- prezeranie výstupov z agenta (logy) v danom čase
- kontrola kamery: pevná alebo napojená na objekt
- „motion blur“ pre indikáciu pohybu robota
- podpora tvorby a analýzy modelu robota
- všetky objekty sveta sú dostupné prostredníctvom stromu objektov na obrazovke
- nezávislosť na platforme

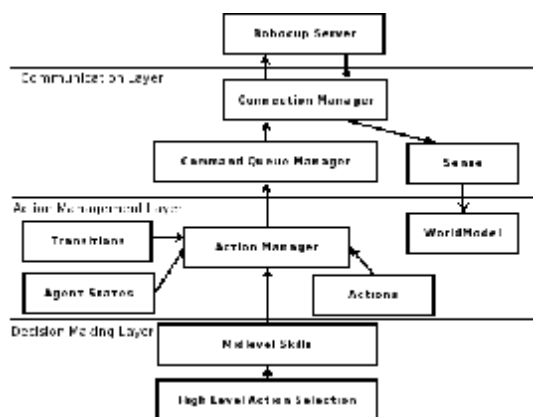
Ako vidno zo schémy agenta tento už pravdepodobne okrem základných pohybov dokázal vykonávať aj vyššie funkcie, o čom nasvedčuje aj fakt že sa tento tým umiestnil v Atlante na treťom mieste.

2.3.3 UI-AI [10]

Pohybový model robota a jeho základné schopností boli definované deterministickým konečným automatom. Jeden stav automatu tvorili polohy jednotlivých častí tela a uhly kĺbov. Tento jednoduchý dizajn v sebe však priniesol niekoľko úskalí:

- akcie robota nebolo možné rozdeliť na menšie podakcie ktoré by boli znovupoužiteľné.
- Robot nebol schopný opustiť vykonávanú akciu uprostred a začať vykonávať inú.
- Agent sa nebol schopný okamžite prispôbiť aktuálnym údajom zo senzorov.

Architektúra agenta bola rozdelená na 3 vrstvy



Obr. 16 Komunikačná vrstva.

Communication layer - Vrstva sa stará o komunikáciu so serverom. Beží paralelne spolu s ostatným

kódom. S každou prichádzajúcou správou aktualizuje svet robota. Ak akčná vrstva vyžaduje vykonanie príkazov táto vrstva zabezpečí ich vhodné usporiadanie v rade príkazov a následné postupné odosielanie serveru na spracovanie.

Akčná vrstva. Táto vrstva zabezpečuje základné zručnosti agenta ako chodenie, otáčanie sa, kopanie a pod. Je rozdelená na štyri časti:

- *Stavy agenta:* Tu sú popísané stavy agenta. Každý stav môže byť opísaný buď veľmi konkrétne ako napr. polohami jednotlivých častí tela, alebo môže byť opísaný aj abstraktne formou podmienok ohraničujúcich určité hodnoty.
- *Prechody:* definujú prechody medzi stavmi. Jeden prechod medzi stavmi jeden a dva znamená že robot sa vie dostať zo stavu jeden do stavu dva.
- *Akcie:* Obsahujú základné schopnosti robota ako chodenie , otáčanie sa, postavenie sa a kopanie. Každá akcia je definovaná ako určitá cesta po množine stavov. Tieto stavy musia medzi sebou mať definované prechody. Pre opakujúce sa pohyby sa využíva opakujúci sa cyklus.
- *Manažér akcií:* Stará sa o prepínanie akcií medzi sebou. V súčasnej verzii nie je možné okamžite sa prepnúť na inú akciu ak sa už vykonáva iná akcia.

Ďalšou vrstvou je *Vrstva rozhodnutí*. Implementuje stredné schopnosti agenta a stará sa o radenie akcií za sebou tak aby sa dosiahol vyšší celok. Príkladom sú funkcie kopni-loptu-smerom, chod'-na-miesto, chyt'-loptu a pod.

Agent disponoval nasledujúcimi schopnosťami:

- *Postavenie sa.* Agent využíva ľudský spôsob postavenia sa. Táto metóda fungovala primerane pre všetky náhodne vygenerované polohy.
- *Chôdza.* Na základe metódy určenia a udržiavania ťažiska tím určil tri základné kroky algoritmu pre chôdzu:
 - *nakláňanie sa do strán* – agent prenesie ťažisko na podpornú nohu, odrazením sa od zeme chodidlom druhej nohy.
 - *posunutie nohy dopredu* – druhá noha sa posunie vpred o určenú dĺžku.
 - *posunutie tela* – ťažisko agenta sa po minulom kroku posunie a agent tak musí pohnúť celou vrchnou časťou tela na vyváženie

Táto metóda napriek svojej stabilite je veľmi pomalá.

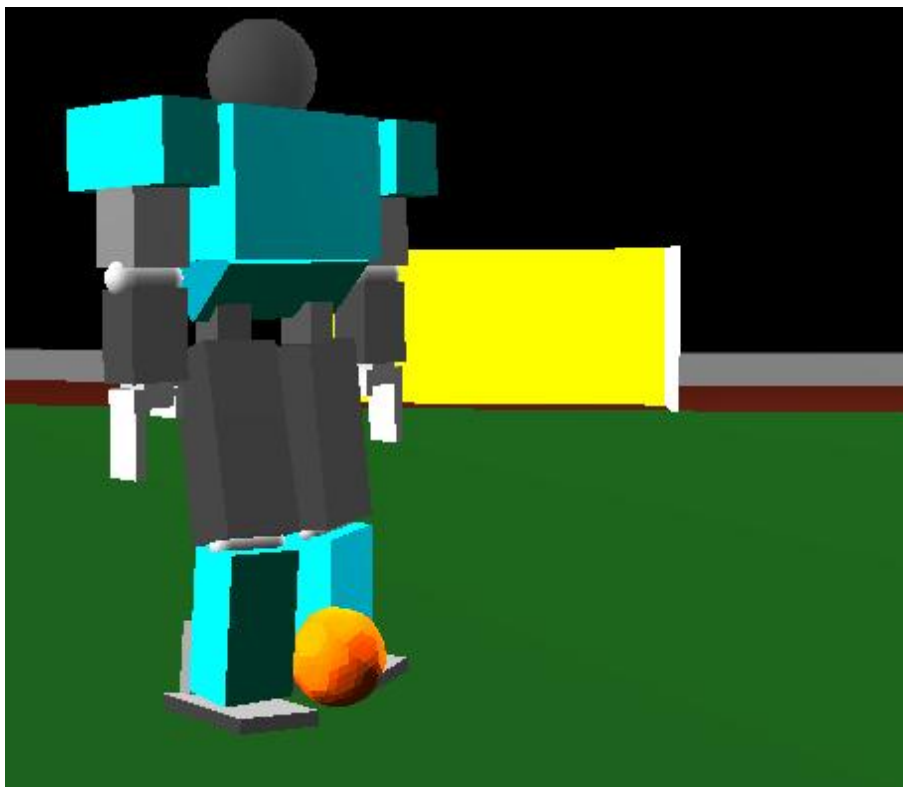
Vývojový nástroj

Tím si vyvinul vlastný nástroj na ovládanie robota za pomoci klávesnice. Bolo možné jednoducho nastaviť ohyby všetkých kĺbov a rýchlosť pohybu. Pomocou tohto nástroja potom vytvorili stavy ktoré boli obsiahnuté v množine stavov v akčnej vrstve agenta ako aj prechodov medzi nimi.

2.3.4 Agent Zigorat [6]

Skupina Zigorat je výsledkom spolupráce viacerých univerzít. Cieľom celého projektu je kooperácia výskumu robotiky medzi univerzitami. Ideou je vytvoriť multi-agentový systém pre výskum širokého spektra algoritmov umelej inteligencií.

Hráč Zigorat je modelom „soccerbot056.rsg“.



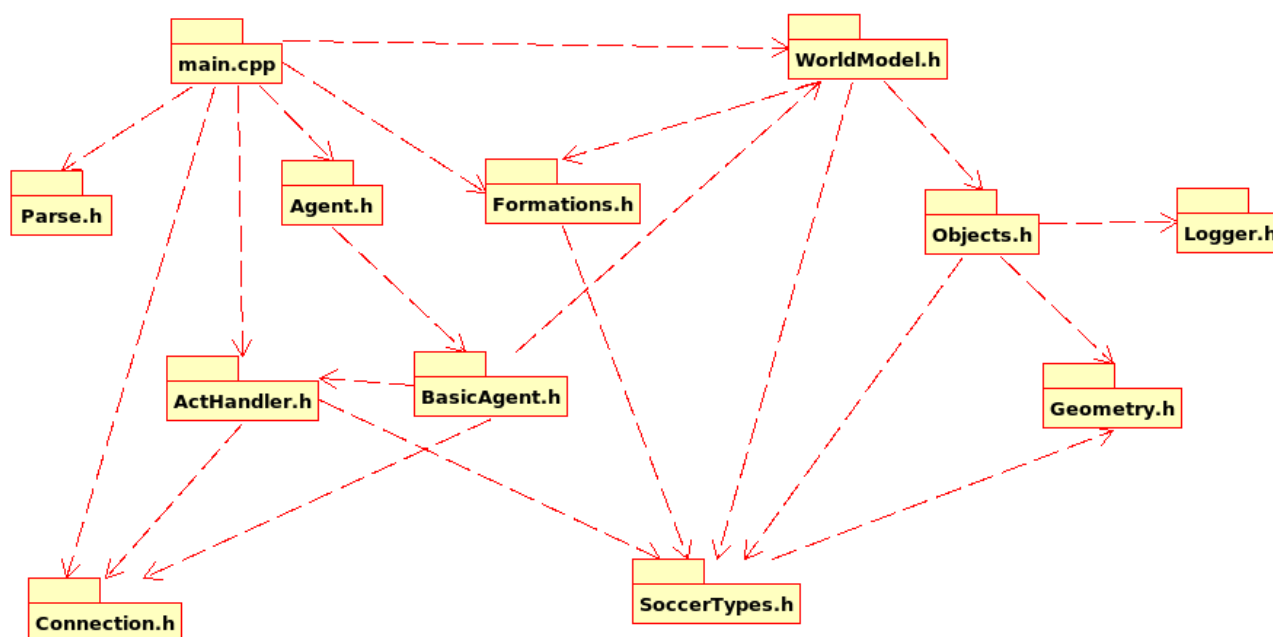
Obr. 17 Model hráča soccerbot056

Na Obr. 18 je znázornené poprepájanie zdrojového kódu na úrovni súborov. Prezeranie začína v súbore `main.cpp`, ktorý obsahuje `main` funkciu aplikácie. Tu sa deklarujú všetky potrebné globálne objekty tried:

- *TroboCupConnection* [*Connection.h*] – poskytuje nadviazanie klienta na server pomocou soketov cez TCP/IP protokol. Prednastavená IP servera je localhost a 3100 port.
- *Formations* [*Formations.h*] – slúži na určenie rozostavenia hráčov v poli. Rôzne formácie ako inicializačná, ofenzívna, defenzívna apod. sú zaznamenané v súbore *formations.conf*. Každý hráč zastupuje určitú rolu v rámci formácie, ktorá je určená v konfiguračnom súbore formácií a označená číslom (brankár, obranca, útočník).
- *WorldModel* [*WorldModel.h*] – objekt tejto triedy poskytuje agentovi všetky potrebné informácie o svojom okolí, rozmery ihriska, bránky, aký je aktuálny čas simulácie, pozície rohových zastávok, informácie o kľbových spojoch agenta, jeho energiu a pod.

- *ActHandler* [*ActHandler.h*] – slúži na zasielanie úkonov / príkazov na server. Úkon je určitá abstrakcia, zovšeobecnenie. Trieda dané úkony pred zaslaním na server prepíše do syntaxe príkazu. Obsahuje zásobník úkonov, ktorý ich akumuluje a jedným príkazom sa odošlú všetky na server. Je možné zasielať príkazy aj úkony na server osobitne.

Agent [*Agent.h*] – objekt triedy *Agent* informuje server, aký model agenta využíva, pracuje s telom agenta ovládaním jeho spojov, t.j. vkladá jednotlivé úkony do zásobníka, ktoré sa odošlú naraz na server. Tu sa aj nachádzajú metódy rôznych akcií, jedna z nich je aj chôdza riešená Zigorat agentom. Podstatnou metódou je *mainLoop*, ktorá vytvára cyklus ukončovaný správou zo servera pri ukončení simulácie. V cykle prebieha rozhodovanie sa agenta na základe prijatých nových informácií prostredia zo servera (ktoré rozparsuje a uloží do svojich lokálnych štruktúr), naplánovanie úkonov do zásobníka a ich odoslanie na server.



Obr. 18 Poprepájanie zdrojového kódu na úrovni súborov

Chôdza Zigorat agenta

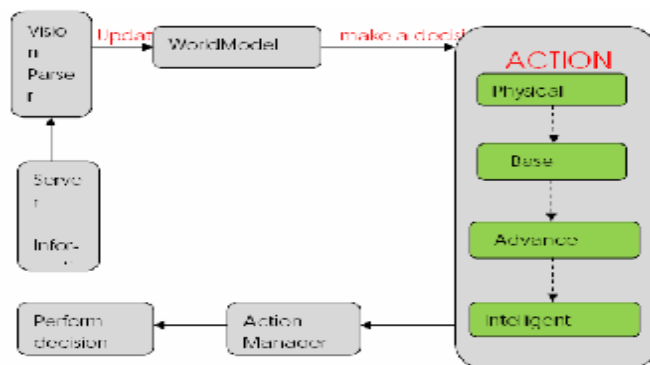
Je definovaná staticky, agent slepo vykonáva úkony bez akejkoľvek dynamickej korekcie. Úkony tela, ako zdvihnutie kolena, posunutie ruky apod., sú plánované podľa relatívneho času simulácie, čo je problém ak príkazy neprijal server v danom potrebnom simulačnom čase. Postupnosť úkonov sa tak naruší a agent padá. Ďalej ak agent narazí na nečakanú prekážku, chôdza sa neprispôsobí na vyrovnanie rovnováhy. Takto určená chôdza je vhodná a úspešná pri bezchybnom spojení a priamej ceste bez prekážok, kde tím agenta Zigorat našli vhodnú postupnosť správne určených úkonov tela pre priamočiary pohyb agenta.

Zigorat sa umiestnil na druhom mieste v robocup3d na turnaji v Atlante a v Teheráne na RoboCup IranOpen 2007 sa dostal až do finále.

2.3.5 DNU Explorer [7]

Hráč sa skladá z niekoľkých častí:

1. *Vision Parser*: Pre transformovanie informácií získaných zo serveru na informácie použiteľné pre "WorldModel".
2. *WorldModel*: Používa informácie z "Vision Parser" pre zistenie aktuálnych informácií o agentovi a hráčom poli, a na podľa nich robí rozhodnutia.
3. *Action Manager*: Dáva konečné rozhodnutia podľa priority jednotlivých akcií, a či sa majú vykonať alebo nie.
4. *Action Perform*: Predá konečné rozhodnutia do vykonávacej sekvencie na vykonanie.



Obr. 19 Schéma agenta DNU Explorer

Action Manager:

3. 1. vytvorí sekvenciu akcií - izoluje každú akciu na detailné kroky a vytvorí z nich sekvenciu
4. 2. filtruje akcie
 - a. posúdi, či sa má akcia vykonať alebo nie. Ak áno vyprázdni sa *vykonávacia sekvencia* a vloží sa do nej daná akcia. Rozhodnutie sa vykonáva podľa priority akcie.
 - b. ak je výsledok predošlého kroku neúspešný, skontroluje sa *vykonávacia sekvencia*, či je prázdna alebo nie. Ak je prázdna vloží sa do nej daná akcia, v opačnom prípade sa vykonajú všetky akcie z *vykonávacej sekvencie* a potom sa do nej vloží daná akcia.
3. vykoná všetky akcie z *vykonávacej sekvencie*

4.1.1 Fantasia [8]

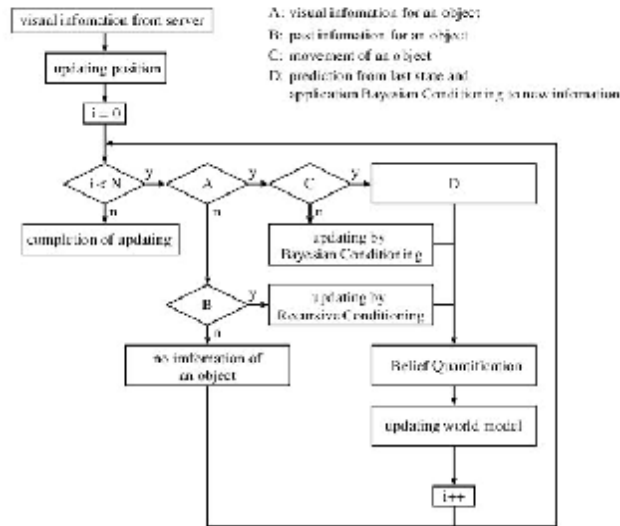
Hráč Fantasia sa zúčastnil RoboCup China Open 2006, kde vyhral prvé miesto v kategórií RoboCup simulation 3D. Využíva koncept *Zero Moment Point* (ZMP) – bod nulového pohybu pre zabezpečenie dynamickej stability dvojnohého robota. ZMP je definovaný ako bod na podlahe, v ktorom je suma všetkých momentov síl je rovná nule. Ak je ZMP vo vnútri konvexného obalu všetkých dotykových bodov medzi chodidlami a podlahou, je možné aby dvojnohý robot chodil. Ďalej sa tento konvexný obal všetkých kontaktných bodov volaný *stabilná oblasť*.

Robot je ovládaný zmenou uhlovej rýchlosti každého kĺbu. Ovládame ho pomocou kľúčových krokov (key frames):

Kde T_m znamená čas trvania m-tého kroku, T_{mn} znamená uhol n-tého kĺbu v m-tého kroku. Kroky medzi týmito krokmi sú približne určené pojitou funkciou. Uhly kĺbov a dobu trvania korok môžu byť ďalej optimalizované pomocou EDA algoritmu (Estimation of Distribution Algorithm).

4.1.2 Naito Striker [9]

Pri tvorbe tohto hráča sa jeho tvorcovia zamerali na vybudovanie akéhosi druhu databázy tak, aby autonómny agent mohol správne vyhodnotiť informácie o ostatných hráčoch a lopte. Používajú vylepšený M. Saxena a T. Guptaov model:



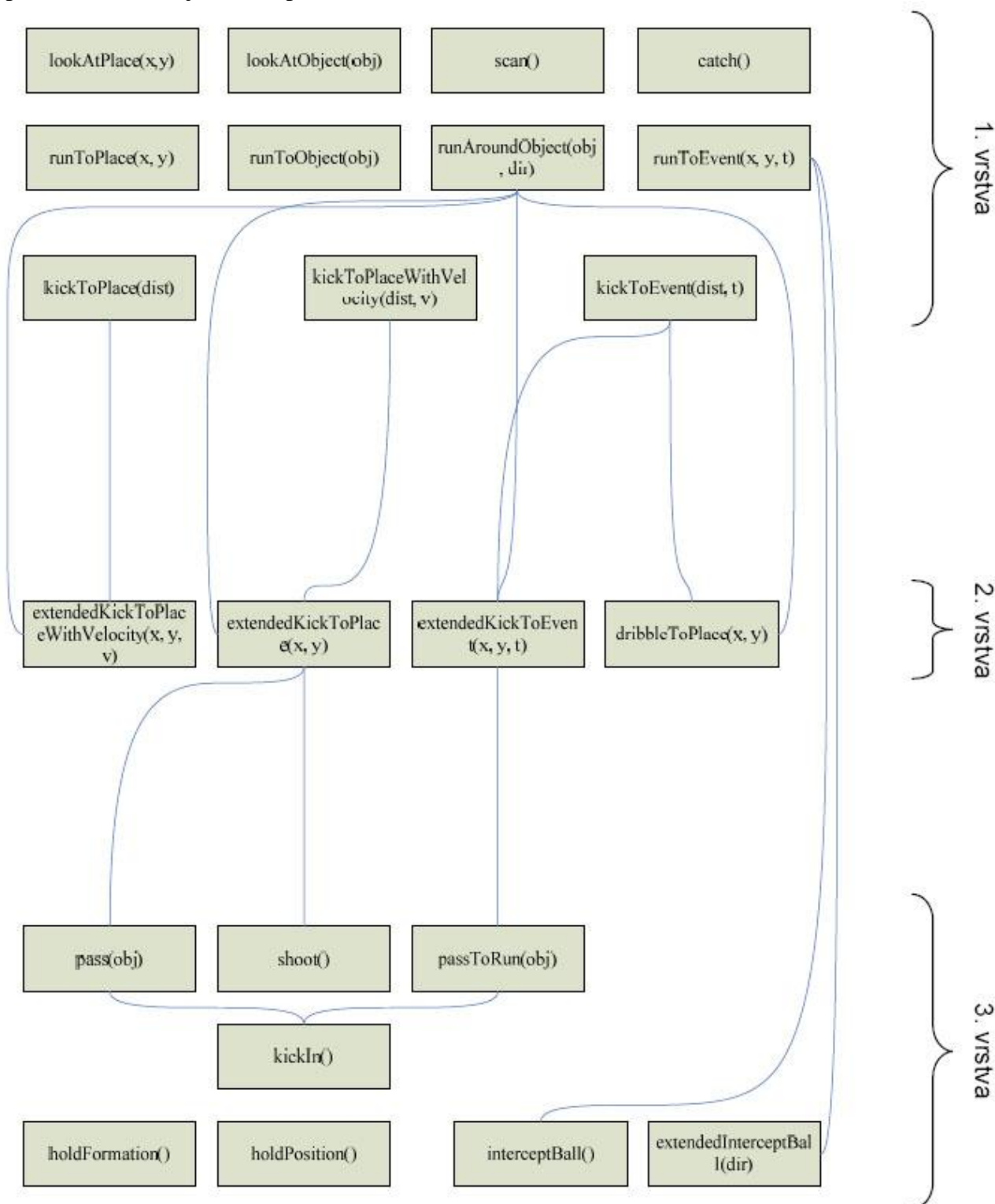
Obr. 20 Schéma agenta Naito Striker

4.1.3 Agent Hazard

Jedná sa o hráča z roku 2006 vytvorenom na predmete Tvorba softvérového systému v tíme. Cieľom bolo vytvoriť hráča na základe práce tímu z roku 2005 s pokročilejšími schopnosťami a vyspelejším rozhodovaním.

Moduly správania

Správanie sa hráča je robené prostredníctvom modulov a rozdelené do troch vrstiev na obr. 21.



Obr. 21 Návrh hierarchie modulov správania

Pre moduly správania vrstvy n platí, že využívajú iba moduly správania nižších vrstiev (t.j. $1 \dots n-1$). Takéto rozdelenie je výhodné z hľadiska vyjadrenia závislostí.

Popis jednotlivých modulov:

Moduly správania prvej vrstvy	
Medzi základné moduly správania patria:	
<ul style="list-style-type: none"> • moduly pre beh hráča • moduly pre kopanie • moduly pre pozorovanie prostredia 	
<i>runToPlace(x, y)</i>	Beh na miesto (x,y)
<i>runToObject(obj)</i>	Beh k objektu obj.
<i>runAroundObject(obj, dir)</i>	Beh k objektu obj. Pri dobehnutí k objektu má mať spojnicu ťažísk objektu a hráča smerový uhol dir.
<i>runToEvent(x, y, t)</i>	Dobehnutie na miesto (x,y) v case t.
<i>kickToPlace(dist)</i>	Kopnutie na vzdialenosť danú parametrom dist. Kope sa smerom v ktorom je hráč natočený.
- <i>kickToEvent(dist, t)</i>	Kopnutie na vzdialenosť danú parametrom dist. Kope sa smerom v ktorom je hráč natočený. Lopta má doraziť na cieľové miesto v čase t.
<i>kickToPlaceWithVelocity(dist, v)</i>	Kopnutie na vzdialenosť danú parametrom dist. Kope sa smerom v ktorom je hráč natočený. Lopta má doraziť na cieľové miesto s rýchlosťou v.
<i>lookAtPlace(x,y)</i>	Hráč natočí kameru tak, aby zadané miesto bolo v strede jeho zorného pola.
<i>lookAtObject(obj)</i>	Hráč natočí kameru tak, aby zadaný objekt bol v strede jeho zorného pola.
- <i>scan()</i>	Hráč otáča kameru tak, aby postupne zaznamenal celé svoje okolie.

Tab. 2 Moduly správania prvej vrstvy

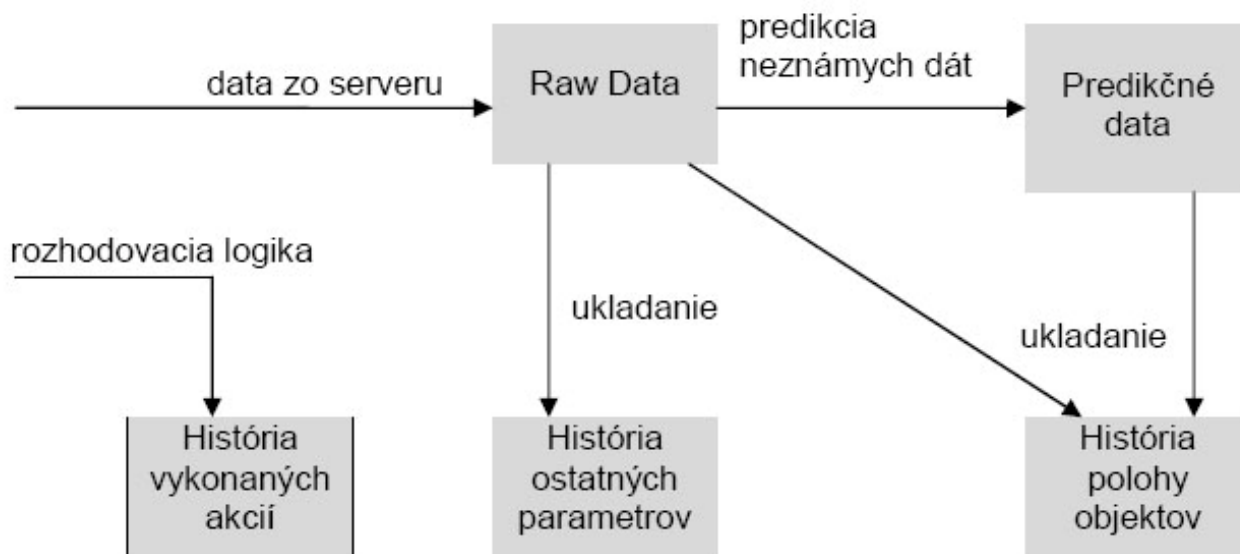
Moduly správania druhej vrstvy	
Medzi rozšírené moduly správania patria rozšírené moduly kopania a modul driblovania s loptou.	
<i>extendedKickToPlace(x, y)</i>	runAroundObject + kickToPlace
<i>extendedKickToEvent(x, y, t)</i>	runAroundObject + kickToEvent
<i>extendedKickToPlaceWithVelocity(x, y, v)</i>	runAroundObject + kickToPlace
<i>dribbleToPlace(x, y)</i>	Beh s loptou na dané miesto. Hráč musí mať pred začiatkom tohto správania loptu vo svojej blízkosti.

Tab. 3 Moduly správania druhej vrstvy

Moduly správania tretej vrstvy	
<i>pass(obj)</i>	Prihrávka objektu obj. Hráč musí mať pred začiatkom tohto správania loptu vo svojej blízkosti
<i>passToRun(obj)</i>	Prihrávka do pohybu objektu obj. Hráč musí mať pred začiatkom tohto správania loptu vo svojej blízkosti.
- <i>shoot()</i>	Strela na bránu. Hráč musí mať pred začiatkom tohto správania loptu vo svojej blízkosti. <i>holdPosition()</i> - Hráč zostáva stať na mieste
<i>holdFormation()</i>	Hráč sa presunie na svoje miesto vo formácii.
<i>kickIn()</i>	Vkopnutie lopty do hry
<i>interceptBall()</i>	Prerušenie pohybu lopty.
<i>extendedInterceptBall(dir)</i>	- Prerušenie pohybu lopty a nastavenie sa do polohy, pri ktorej má spojnice lopty a hráča uhol dir.

Tab. 4 Moduly správania tretej vrstvy

Agentov model sveta je reprezentovaný tzv. *information storage* znázornenom na obrázku



Obr. 22 Information Storage

Raw data – je vrstva, ktorá obsahuje a pracuje s informáciami, ktoré prichádzajú zo servera a sú (odhliadnuc od možného šumu) presné.

Predikčné dáta – sú generované prediktorom, sú nevyhnutné na tvorbu stratégie.

História polohy objektov – v tejto časti sú uložené všetky súradnice objektov od začiatku simulácie, sú to informácie o polohe v polárnych súradniciach ako prichádzajú zo servera a tiež prepočítané pravouhlé súradnice. Taktiež je tu história predikovaných súradníc.

História vykonaných akcií – všetky akcie vykonané daným hráčom v chronologickom poradí.

História ostatných parametrov – stav batérie, teplota, uhol pohľadu, mód hry.

Hráč ako taký je schopný samostatnej hry. Implementácia z časových dôvodov však nebola kompletná. Niektoré moduly sú vytvorené len na úrovni prototypu.

4.2 Zhodnotenie analýzy

Podrobne sme si rozobrali simulačné prostredie v ktorom prebiehajú zápasy RoboCup-u. Zistili sme si aktuálny model humanoidného robota. Je to pomerne veľký skok od predošlej verzie servera s ktorou pracoval napríklad tím v minulom roku kedy model robota tvorila len guľa. Aktuálny model má humanoidnú formu a disponuje niekoľkými kĺbmi po celom tele, pričom kĺby môžu byť dvoch rôznych druhov. Zistili sme že komunikácia medzi robotmi a serverom prebieha na protokoloch TCP alebo UDP je teda možné (a na súťaži nutné) spustiť svoj tím na osobitnom počítači a po sieti sa pripojiť na počítač kde beží serverová aplikácia. Analýza rôznych tímov ukázala aké rôznorodé riešenia je možné použiť pri tvorba agentov. Spomedzi analyzovaných agentov sme si pre ďalšiu prácu vybrali dvoch a to konkrétne agenta Hazarda z minulého roku od tímu 6th sense, a agenta Zigorata. Agent hazard disponuje výhodným modelom správania sa organizovaným do vrstiev z hľadiska úrovne správania sa a ďalej do modulov. Táto architektúra sa javí ako výhodná, flexibilná a ľahko rozšriteľná. Bohužiaľ agent hazard je postavený na staršom type servera a preto nemôžeme jeho kód využiť celý. Preto sme hľadali ďalej a ako už bolo spomenuté ako druhého kandidáta sme si vybrali agenta Zigorata. Tento agent je stavaný na nový typ serveru a poskytuje vynikajúci model komunikácie so serverom. Agent taktiež disponuje základnými pohybovými vlastnosťami ako je otáčanie sa a chôdza tie sa však pri našich testoch javili ako pomerne nestabilné. Rozhodli sme sa použiť oboch agentov a z každého zobrať to najlepšie.

5. Špecifikácia požiadaviek

Cieľom projektu bude navrhnuť prototyp hráča resp. časti hráča. Tento prototyp by mal byť schopný fungovať na novom type servera a mal by mať formu humanoidného robota. Opierať sa budeme o ideu tímu z minulého roka v predmete TSST, ako aj našej analýzy iných svetových tímov. Budeme sa sústreďovať na základné schopnosti hráča. Nový typ serveru ako aj humanoidný hráč majú iné charakteristiky ako ich staršie verzie. Úlohou bude základná orientácia hráča v priestore, jednoduchý pohyb (chôdza) a schopnosť postaviť sa.

6. Hrubý návrh

Z analýzy predošlých agentov sme usúdili, že použitie statických metód na riadenie jednotlivých funkcií agentov nie je vhodné. Z tohto dôvodu sme si zvolili použitie evolučného algoritmu. Keďže táto metóda ešte nebola použitá rozhodli sme sa špecializovať len na jednu oblasť súťaže development a tou je vstávanie agenta z ľubovoľnej polohy.

Z analýzy štruktúr agentov vyplynulo, že najvhodnejšie bude skombinovať komunikačnú vrstvu ľubovoľného agenta a vytvoriť, respektíve prevziať architektúru modelu správania sa z agenta Hazarda. Tento postup sme zvolili z dôvodu veľmi kvalitnej štruktúry modelov správania sa v agentovi Hazardovi. Aj keď je agent vytvorený na starú generáciu RoboCup serverov (guličky), všeobecná štruktúra jeho modelov správania sa dá aplikovať aj na novú generáciu agentov.

Základnou myšlienkou nášho agenta bude využitie evolučného algoritmu na dosiahnutie cieľa. Naším cieľom je postavenie sa do polohy, ktorá bude vhodná na ďalší pohyb. Treba povedať, že táto poloha nebude určená staticky, ale bude sa odvíjať od celkovej situácie a potrieb, ktoré budú určené vyššou plánovacou vrstvou.

Pre príklad: agent stratí rovnováhu a spadne. Pred spadnutím bolo jeho cieľom priblížiť sa k lopte. Agent zahájí model správania ktorý zabezpečí postavenie, pričom však výsledným cieľom tohto kroku nebude statické rozloženie častí tela, ale taká poloha, z ktorej sa dá najefektívnejšie pokračovať v nadradenom pláne, t.j. poloha z ktorej agent najlepšie zahájí pohyb k lopte.

6.1 Štruktúra agenta

Agent neurotik sa bude skladať zo štyroch hlavných častí:

1. Hlavný vykonávací cyklus
2. Komunikačný modul
3. Pohľad na svet (worldview modul)
4. Modely správania sa

Hlavný vykonávací cyklus bude samotné jadro agenta. V tejto časti agenta sa bude vykonávať volanie jednotlivých podmodulov ktoré zaručia danú funkčnosť.

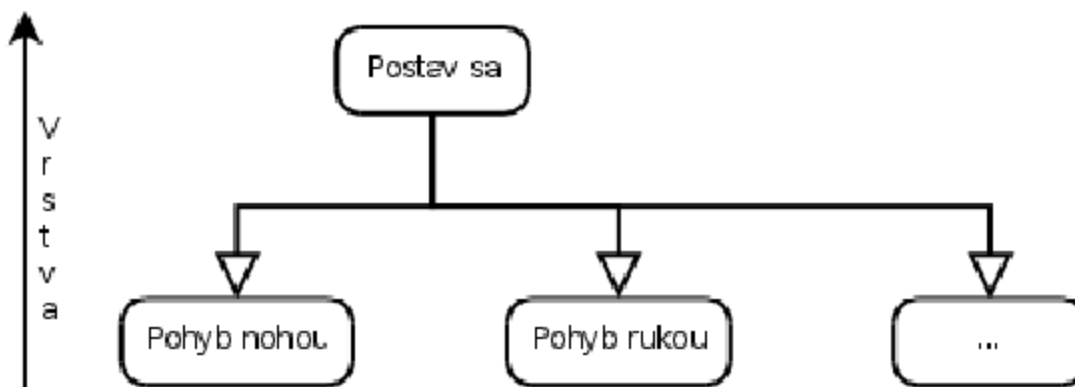
Komunikačný modul agenta bude prevzatý z agenta Zigorata. Modul bude upravený pre naše potreby, hlavne plánujeme zmeny v rozhraní, funkčnosť ostatne nezmenená. Modul sa bude starať o posielanie a prijímanie správ zo servera.

Pohľad na svet sa prevezme z agenta Zigorata. Pre naše potreby budú zmeny v tejto časti minimálne.

Agent Neurotik bude využívať štruktúru modelov správania sa (behaviors) z agenta Hazarda. Štruktúra je založená na modularite a jednoduchosti jednotlivých modelov.

Každý model je reprezentovaný triedou, ktorá je zdedená z abstraktného rozhrania BaseBehaviorModule. Toto rozhranie obsahuje metódy, ktoré sa v konkrétnych modeloch implementujú. Pomocou makier REGISTER_MODULE(meno) sa jednotlivé moduly zaregistrujú do systému. Makrom USE_MODULE(meno, cesta) sa modul prihlási ako aktívny na použitie.

Samotné moduly sú rozdelené do viacerých vrstiev. Agent Hazard mal 3 vrstvy, pričom platí, že každý model vrstvy x , môžu používať len moduly vrstvy $x - 1$.



Obr. 23 Vrstvy modelov udalostí

V našom agentovi tento princíp zachováme, avšak počet vrstiev modelov sme obmedzili na dve. Spodná vrstva bude obsahovať základné modely pre pohyb jednotlivých častí tela, a vrchná vrstva bude obsahovať primitívne modely typu „postav sa“. Systém bude navrhnutý tak, aby bol do budúcnosti rozšíriteľný.

6.2 Model vstávania

Hlavnou úlohou agenta Neurotika v tomto projekte je postavenie sa do vhodnej polohy vzhľadom na celkový plán akcie na vyšších úrovniach. Z analýz sme usúdili, že statický prístup k problémom rovnováhy nie je vhodný, zvlášť pre situáciu postavenia kde nie je možné spoliehať na predpoklady určitej začiatočnej polohy.

Z tohto dôvodu sme si vybrali ako spôsob riešenia použitie evolučného algoritmu.

6.3 Využitie evolučných algoritmov

Evolučné algoritmy využívajú, na riešenie optimalizačných problémov, princípy evolúcie živej hmoty. Simulujú darwinov evolučný proces. Ten obsahuje tieto tri zložky [12]:

1. **Prirodzený výber** – Silnejší jedinci majú väčšiu šancu reprodukcie ako slabší jedinci.
2. **Náhodný genetický drift** – Náhodné udalosti v živote jedincov. Napríklad náhodná smrť silného jedinca pred tým, než dostal šancu na reprodukciu.
3. **Reprodukcia** – Genetická informácia potomkov vzniká kombináciou génov rodičov, pričom môže dochádzať k jej poškodeniu (mutácii).

Simulovaná evolúcia prebieha nad určitou množinou jedincov, v ktorej sa cyklicky opakuje výber rodičov a reprodukcia. Po reprodukcii vzniká nová množina jedincov a evolúcia ďalej prebieha nad touto novou množinou.

Takouto simulovanou evolúciou môžeme napríklad naučiť hráča pohybovať sa. Pohyb hráča je v podstate zložený z množstva pohybov jeho motorčekov. Celý pohyb sa teda dá definovať ako chronologická postupnosť rýchlostí otáčania všetkých hráčových motorčekov. Na začiatok by bolo

vhodné otestovať možnosti aplikácie niektorého z evolučných algoritmov na jednoduchšom pohybe. Napríklad na postavení sa.

Postavenie sa pomocou evolučného algoritmu

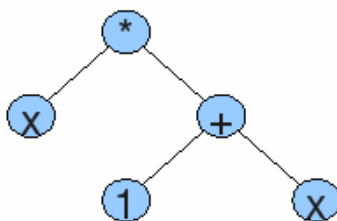
Pre realizáciu simulovanej evolúcie je potrebné zadať nasledovné:

- jedinca a jeho reprezentáciu
- spôsob ohodnocovania schopnosti jedincov prežiť (fitness)
- spôsob výberu rodičov
- operácie kríženia a mutácie
- spôsob výberu jedincov do novej generácie

6.3.1 Reprezentácia jedinca

Jedinec je definovaný genetickou informáciou. Tá musí (pre potreby evolúcie pohybu) definovať pohyb robota. Bude ju teda tvoriť chronologická postupnosť základných pohybov robota (pohybov jednotlivých kĺbov), indexovaných časom, kedy sa tento pohyb začne vykonávať. Dĺžka chromozómu nie je dopredu známa. Preto je vhodné použiť tzv. „Messy“-chromozómy [13] (m-chromozómy). Tie majú premenlivú dĺžku. Samotný chromozóm je tvorený postupnosťou usporiadaných dvojíc, zložených z indexu a hodnoty génu. Ak sa v chromozóme nachádza viac génov s rovnakým indexom, použije sa len jeden z nich, a to prvý v poradí. V tomto konkrétnom prípade by index tvoril čas a hodnotu génu zoznam vstupov pre metódy zabezpečujúce pohyb jednotlivých kĺbov.

Robot má senzory, ktoré mu poskytujú nejaké informácie o okolitom priestore. Bolo by teda vhodné, aby robot riadil svoj pohyb na základe týchto informácií. Pohyby by teda mali byť relatívne vzhľadom na tieto informácie (aspoň na niektoré z nich). Ale ako? Dopredu určiť túto závislosť je nemožné. Mohla by teda tiež byť predmetom evolúcie. Elementy zoznamu hodnôt jednotlivých génov by nemali byť len obyčajné čísla. Mohli by to byť funkcie, ktoré umožnia ich výpočet na základe robotových informácií o okolí. Problémom tohto typu sa zaoberá metóda *Genetického programovania* [13]. V nej je gén tvorený koreňovým stromom, kde sú vrcholy stromu ohodnotené funkciou a listy hodnotami.



Obr. 24 Príklad koreňového stromu. Funkcia $F(x)=x*(1+x)$

Jedinec teda bude reprezentovaný m-chromozómom, v ktorého génoch bude index reprezentovať čas a hodnotu zoznam koreňových stromov.

6.3.2 Fitness

Fitness označuje schopnosť prežitia a reprodukcie jedincov. Spôsob ohodnocovania fitness teda určuje kam bude evolúcia smerovať. V tomto prípade chceme aby sa robot čo najrýchlejšie postavil tak, aby mohol čo najskôr vykonať inú akciu (musí stáť na zemi) a aby pri tom minul čo najmenej energie. Do výpočtu fitness teda vstupujú nasledovné parametre:

1. Rozdiel maximálnej dosiahnutej výšky hlavy a výšky hlavy vo vzpriamenej polohe
2. Čas za ktorý sa do tejto výšky dostal ako dlho v nej zotrval (či potom nespadol)
3. Množstvo energie potrebnej na postavenie sa.

Parameter číslo jedna je rozdiel z dôvodu zamedzenia vyskakovania do čo najvyššej výšky.

Presné závislosti a podiely jednotlivých parametrov sa nedajú dopredu presne určiť a budú určené pri testovaní a ladení.

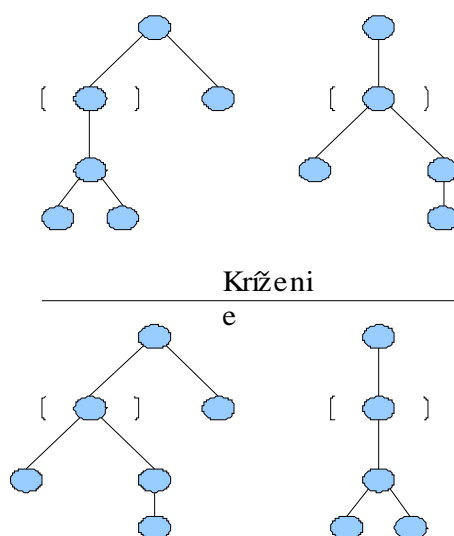
6.3.3 Spôsob výberu rodičov

Po ohodnotení jedincov hodnotou fitness, je nutné vybrať niekoľko jedincov, ktorý pristúpia k reprodukcii. Stratégií ne tento výber je niekoľko. Napríklad výber najlepších jedincov, turnajový výber, pseudonáhodný výber, ... No pseudonáhodný výber imituje aj náhodne udalosti v živote jedincov. Preto budeme používať túto stratégiu, konkrétne implementovanú pomocou ruletového výberu.

6.3.4 Operácie kríženia a mutácie

Operácie kríženia musia byť definované tak, aby zasahovali chromozóm aj jednotlivé gény v chromozóme. Začneme teda tieto operácie definovať od najnižších vrstiev.

Kríženie dvoch koreňových stromov spočíva vo výmene podstromov. V oboch stromoch sa nezávisle od seba náhodne zvolí bod kríženia. A podstromy začínajúce v týchto bodoch kríženia sa jednoducho vymenia.



Obr. 25 Príklad kríženia dvoch stromov. Zátvorky označujú náhodne zvolené body kríženia.

Mutáciou stromu budeme rozumieť nahradenie náhodného podstromu iným, náhodne vygenerovaným, stromom. Kríženie dvoch génov bude spočívať v postupnom krížení všetkých ich

stromov na zhodných indexoch (prvý s prvým, druhý s druhým, ...) a výberu indexu z jedného z nich. Mutácia génu je náhodná zmena indexu a mutácia stromov. Kríženie dvoch m-chromozómov potom bude prebiehať nasledovne. Náhodne sa zvolí bod kríženia a začnú sa krížiť jednotlivé gény. Až po bod kríženia chromozómov sa budú brať indexy z prvého rodiča a za bodom kríženia sa budú brať indexy z druhého rodiča. Mutácia m-chromozómu je náhodné pridanie alebo odobratie génu.

6.3.5 Výber jedincov do novej generácie

V novej populácii budú určite všetci potomkovia, ktorí vynikli pomocou kríženia alebo mutácie z predchádzajúcej populácie. No treba sa rozhodnúť či do nej zahrnieme aj niekoľko jedincov z predchádzajúcej populácie. Napríklad najlepších jedincov. Evolučné algoritmy imitujú evolúciu v prírode. Dalo by sa teda napríklad uvažovať o nejakom veku jedincov.

Ak do novej generácie zahrniem niekoľko najlepších jedincov, ich gény nebudú zabudnuté pri nevhodnej voľbe rodičovských párov a zlom výsledku reprodukcie. No jedince by sa mohli vyberať aj ruletovým systémom.

Presný systém výberu spresníme pri testovaní.

7. Podrobný návrh

V tejto kapitole podrobnejšie rozoberáme návrh niektorých častí robota. Našou úlohou je preskúmanie evolučných algoritmov a možnosti ich aplikácie na učenie sa robota. Primárne sa teda zameriame na podrobnejší návrh evolučných algoritmov.

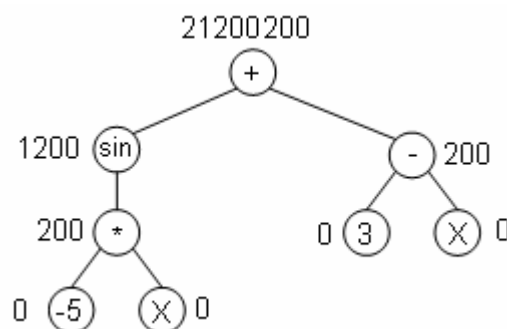
7.1 Návrh evolučného algoritmu

Na začiatku sa náhodne vygeneruje populácia jedincov. Veľkosť populácie sa určí experimentálne. Jedinci sa vykonajú a ohodnotia sa hodnotou fitness. Vyberú sa rodičovské páry, prebehne kríženie a mutácia. Potomkovia sa vykonajú a ohodnotia sa hodnotou fitness. Následne sa pridajú do pôvodnej populácie a z nej sa vytvorí nová populácia. Tým, že sa do novej populácie berú aj jedince zo starej populácie, sa zabráni strate dobrých génov pri zlom výbere rodičov. Ďalej sa už len bude opakovať cyklus od výberu rodičov, po vytvorenie novej generácie.

Pri vytvorení novej generácie sa jedinci uložia do súboru, pre prípad vypnutia počítača a pre umožnenie opätovného prezerania vyevolvovaných jedincov.

7.1.1 Reprezentácia jedinca

Jedinec bude reprezentovaný pomocou m-chromozómu. Ten je tvorený nešpecifikovaným množstvom génov. Gén bude obsahovať identifikátor správania, ktoré reprezentuje. Parametre správania budú tiež zapísané v géne, a to vo forme koreňových stromov. Koreňový strom reprezentuje funkciu. Všetky jeho nelistové vrcholy budú ohodnotené funkciami a listy operandami. Funkcie pre ohodnotenie vrcholov sa budú voliť z nasledujúcej množiny $\{+, -, *, /, \sin, \cos\}$. Táto množina sa počas testovania možno rozšíri o ďalšie funkcie. Operandy budú tvoriť konštanty, alebo premenné z hráčovho modelu sveta. Príklad koreňového stromu je na nasledujúcom obrázku .



Obr. 26 Koreňový strom s Readovým lineárnym kódom každého podstromu (čísla mimo stromu)

Koreňový strom bude reprezentovaný Readovým lineárnym kódom a vektorom ohodnotení vrcholov. Readov lineárny kód tvorí počet potomkov stromu a kódy potomkov. Napríklad strom na obrázku (**Chyba! Nenašiel sa žiaden zdroj odkazov.**) má kód 21200200. Prvé číslo je počet potomkov koreňa (2), nasledujú kódy potomkov (1200 a 200). Počet koreňových stromov závisí na množstve parametrov konkrétneho behavioru. Jednotlivé typy génov sa teda budú líšiť. Vo všeobecnosti bude gén obsahovať čas, identifikátor správania a parametre správania (vo forme koreňových stromov).

7.1.2 Fitness

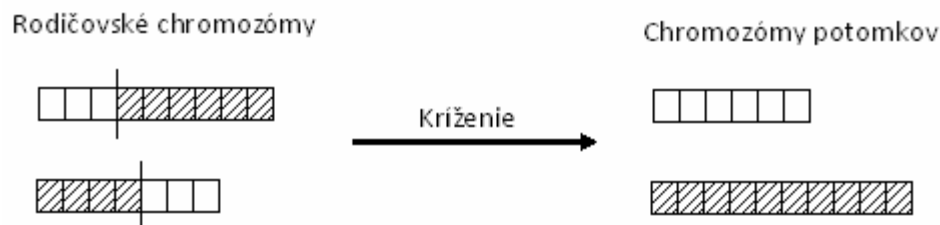
Funkcia fitness bude závislá od priblíženia sa ku očakávanej polohe hráča (pri učení pohybu), spotrebovanej energii a času, za ktorý sa tam dostal. Pravdepodobne to bude váhovaný súčet týchto parametrov.

7.1.3 Spôsob výberu rodičov

Rodičovské páry sa budú vyberať pomocou rulety podľa hodnoty fitness. Tento spôsob výberu simuluje náhodné udalosti v živote jedincov. Počet vybraných jedincov sa určí experimentovaním.

7.1.4 Operácie kríženia a mutácie

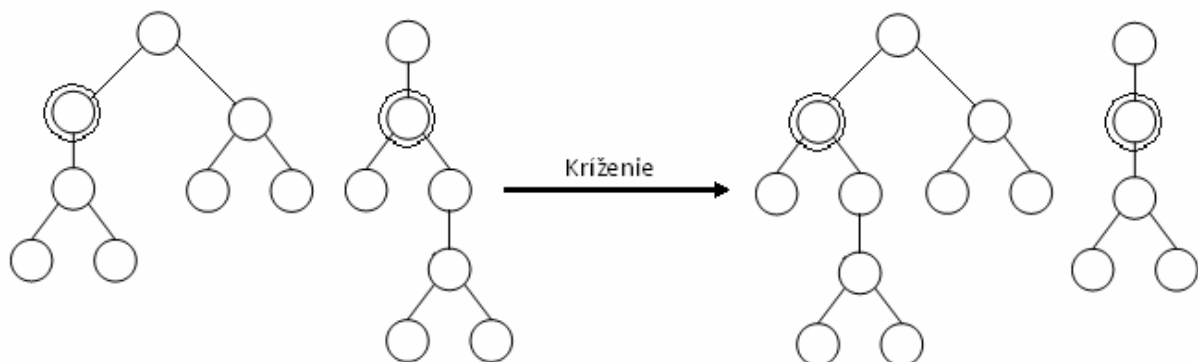
Dvaja jedinci sa budú krížiť jednoduchým jednobodovým krížením m -chromozómov. To znamená, že sa zvolia dva náhodné body kríženia (jeden v každom jedincovi) a gény za týmito bodmi sa vymenia. Mutácia jedinca bude prídanie alebo odoberanie génov.



Obr. 27 Kríženie dvoch m -chromozómov

Pri krížení jedincov sa vyberú aj náhodné gény z jedného rodiča. K nim sa následne nájdu gény rovnakého typu z druhého rodiča. Ak druhý rodič obsahuje takéto gény prebehne kríženie aj na nich. Gény sa budú krížiť tak že sa skrížia ich stromy. Krížiť sa budú stromy s rovnakým indexom. Teda tie, ktoré reprezentujú rovnaký parameter správania. Mutácia génu bude mutácia jeho stromov a zmena času tohto génu.

Kríženie koreňových stromov spočíva vo výmene ich podstromov. Vyberú sa teda dva podstromy (z každého stromu jeden) a tie sa navzájom vymenia (**Chyba! Nenašiel sa žiaden zdroj odkazov.**). Mutácia stromu bude výmena náhodného podstromu za nový náhodne vygenerovaný strom.



Obr. 28 Kríženie dvoch koreňových stromov

7.1.5 Výber jedincov do novej generácie

Jedince do novej generácie sa budú vyberať ruletou podľa hodnoty fitness, alebo sa vyberú najlepšie jedince. Ruletový výber simuluje náhodné udalosti. No na druhú stranu výber najlepších jedincov zabráni strate dobrých génov zlým výberom.

7.1.6 Vykonanie jedinca

Vykonanie jedinca znamená postupné vykonanie pohybov, ktoré sú v jeho génoch. Každý gén jedinca reprezentuje určité správanie (určitý pohyb) v čase. Takže v určitom čase sa musia vykonať pohyby určené pre tento čas. Ak jedinec obsahuje viacero génov s rovnakým časom a správaním, vykoná sa len prvý z nich.

Jedince sa budú vykonávať sériovo a medzi vykonaním dvoch jedincov sa hráč odpojí a pripojí k serveru. Paralelizmus vykonávania hráčov spočiatku nebude možný, aby sa zabránilo interakcii hráčov. No v neskorších fázach by interakcia mohla pomôcť. Hráči by totiž mali reagovať na okolie a hlavne by to urýchlilo evolúciu.

Po vykonaní jedinca sa vypočíta jeho fitness.

8. Prototyp

Ako prototyp sme zvolili vytvorenie jednoduchého agenta, ktorý bude vedieť komunikovať zo serverom a ukázať funkčnosť modelov udalostí. Jeho modely nebudú ešte spĺňať užitočnú funkciu.

Ciele:

1. Prvým cieľom je preukázať, že nami zvolená kombinácia častí jednotlivých modulov a ich architektúra je implementovateľná.
2. Ďalším cieľom je overenie správnosti komunikačnej vrstvy prevzatej z agenta Zigorata. Treba overiť, či je vrstva kompletná, teda či obsahuje spracovanie všetkých možných príkazov a správ.
3. Posledným cieľom je otestovanie systému modelov udalostí prevzatého z agenta Hazarda, či je dostatočný pre nové prostredie a jeho požiadavky, alebo potrebuje rozšírenia.

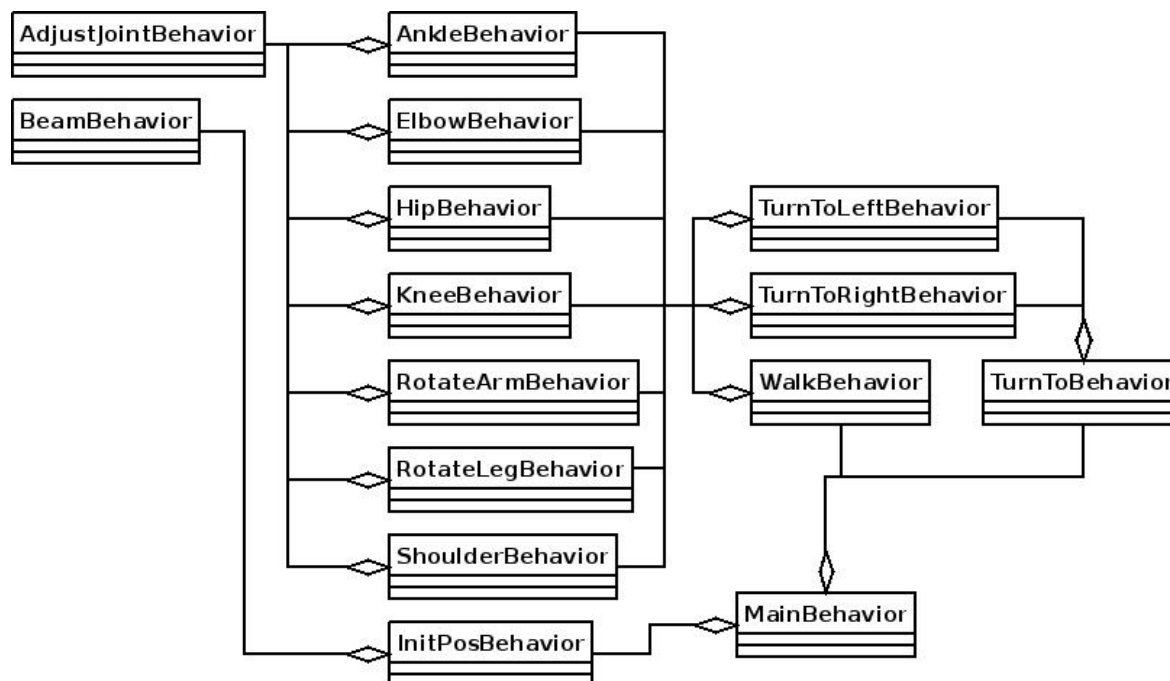
8.1 Inštalácia

K tomuto postupu kompilácie hráča je potrebné mať nainštalované vývojové prostredie Eclipse s podporou pre C++ (v operačnom systéme Ubuntu balíky *eclipse*, *eclipse-cdt*). Najprv je potrebné vytvoriť nový prázdny C++ projekt s názvom *NeuroticAgent*. Eclipse vytvorí k novému projektu aj rovnomenný adresár, ktorý nájdeme vo *workspace* adresári v ktorom sme projekt vytvorili. Do tohto adresára skopírujeme zdrojové kódy hráča. V eclipse, potom stačí zvoliť *Project-> Build Project*. Skompilovaného hráča nájdeme v podadresári projektu, s názvom *Debug*. Pre jeho spustenie je potrebné mať spustený server.

8.2 Štruktúra hráča

Hráč je rozložený do modulov – správání („behaviors“). Tie majú rôzne úrovne a hierarchicky sa navzájom volajú. Správania sú triedy C++. Okrem *BasicBehavior*, má každá z nich má funkciu *behave*, ktorá je volaná pri aplikovaní daného správania.

Prototyp obsahuje základné motorické moduly správania a jeden ukázkový modul z vyššej vrstvy, ktorý vykonáva chodenie (iba staticky naprogramované). Štruktúra previazanosti správaní v prototypu je nasledovná:



Obr. 29 UML diagram štruktúry behaviorov

8.3 Činnosť hráča

Pri spustení hráča sa hráč pripojí k serveru a zavolá sa hlavný modul správania – MainBehavior. Ten potom ďalej rozhoduje, čo sa bude diať. V prototypu sa tento modul snaží otočiť smerom k súperovej bránke a napredovať k nej pomocou modulov TurnToBehavior a WalkBehavior. Tie zas volajú ďalšie moduly pre základné motorické funkcie.

8.4 Popis behaviorov

1. **AdjustJointBehavior** – Podľa zadaných hodnôt priamo nastaví uhlovú rýchlosť kĺbu.
vstup: ID kĺbu a dva uhly určujúce jeho natočenie
výstup: TRUE ak boli zadané hodnoty akceptované, v opačnom prípade FALSE
2. **AnkleBehavior** – Nastaví uhol členku.
vstup: voľba medzi pravou/ľavou nohou, dva uhly určujúce jej natočenie, rýchlosť akou sa zmena má vykonať
výstup: TRUE ak boli zadané hodnoty akceptované, v opačnom prípade FALSE
3. **BasicBehavior** – Základná trieda hráča, z ktorej dedia všetky ostatné triedy, nemá žiadnu špeciálnu funkcionálnosť.
4. **BeamBehavior** – Vysiela *beam* príkaz serveru. *Beam* príkaz preniesie hráča na požadovanú pozíciu (nie počas hry).

- vstup: X a Y súradnice, uhol – smer natočenia hráča
výstup: TRUE ak boli zadané hodnoty akceptované, v opačnom prípade FALSE
5. **ElbowBehavior** – Nastaví uhol laktáa.
vstup: voľba medzi pravou/ľavou rukou, uhol určujúci jej natočenie, rýchlosť akou sa zmena má vykonať
výstup: TRUE ak boli zadané hodnoty akceptované, v opačnom prípade FALSE
6. **HipBehavior** - Nastaví uhol bedrového kĺbu.
vstup: voľba medzi pravou/ľavou nohou, dva uhly určujúce jej natočenie, rýchlosť akou sa zmena má vykonať
výstup: TRUE ak boli zadané hodnoty akceptované, v opačnom prípade FALSE
7. **InitPosBehavior** – Inicializuje hráča a nastaví všetko potrebné pre jeho štart.
8. **KneeBehavior** - Nastaví uhol kolena.
vstup: voľba medzi pravou/ľavou nohou, uhol určujúci jej natočenie, rýchlosť akou sa zmena má vykonať
výstup: TRUE ak boli zadané hodnoty akceptované, v opačnom prípade FALSE
9. **MainBehavior** – Trieda rozhodujúca na základe zložitejších procedúr o nasledujúcej postupnosti akcií. V prototypy táto trieda vykonáva samotné chodenie – postupnosť jednotlivých menších elementárnych akcií súvisiacich s chodením.
10. **RotateArmBehavior** - Nastaví uhol otočenia ramena.
vstup: voľba medzi pravým/ľavým ramenom, uhol určujúci jej natočenie, rýchlosť akou sa zmena má vykonať
výstup: TRUE ak boli zadané hodnoty akceptované, v opačnom prípade FALSE
11. **RotateLegBehavior** - Nastaví uhol otočenia nohy.
vstup: voľba medzi pravou/ľavou nohou, uhol určujúci jej natočenie, rýchlosť akou sa zmena má vykonať
výstup: TRUE ak boli zadané hodnoty akceptované, v opačnom prípade FALSE
12. **ShoulderBehavior** - Nastaví uhol plecom.
vstup: voľba medzi pravým/ľavým plecom, dva uhly určujúce jeho natočenie, rýchlosť akou sa zmena má vykonať
výstup: TRUE ak boli zadané hodnoty akceptované, v opačnom prípade FALSE
13. **TurnToBehavior** – Otočí hráča na požadovaný uhol.
vstup: uhol, o ktorý sa má hráč otočiť
výstup: TRUE ak boli zadané hodnoty akceptované, v opačnom prípade FALSE
14. **TurnToLeftBehavior** – Otočí hráča požadovaným na požadovaný uhol smerom doľava.
vstup: uhol, o ktorý sa má hráč otočiť smerom doľava
výstup: TRUE ak boli zadané hodnoty akceptované, v opačnom prípade FALSE
15. **TurnToRightBehavior** - Otočí hráča požadovaným na požadovaný uhol smerom doprava.
vstup: uhol, o ktorý sa má hráč otočiť smerom doprava
výstup: TRUE ak boli zadané hodnoty akceptované, v opačnom prípade FALSE

16. *WalkBehavior* – Hráč kráča rovno v smere, v ktorom je otočený.

8.5 Záver

Implementáciou a otestovaním prototypu sme overili navrhnuté architektonické riešenie, funkčnosť a možnosť využitia existujúcej implementácie pri práci v ďalšom semestri.

8.5.1 Overenie architektonického návrhu agenta

Prototyp využíva modulárnu architektúru agenta *Hazard* a moduly správania, ktoré využívajú funkčnosť agenta *Zigorat*. Relatívne bezproblémové integrovanie architektúry (*Hazard*) a dostupnej funkcionality (*Zigorat*) indikuje dobrý architektonický návrh agenta.

Najväčšou zmenou oproti modulárnemu systému agenta *Hazard* bolo nahradenie pôvodného spôsobu registrovania modulov správania v systéme na úrovni implementácie. Pôvodné využitie frameworku *Zeitgeist* sme vyhodnotili ako zbytočné, nakoľko ani v agentovi *Hazard* sa jeho využitie zatiaľ neukázalo ako opodstatnené. Využitie jednoduchšej implementácie viedlo k eliminovaniu zbytočných závislostí a zjednodušeniu na úrovni kódu, kompilácie, testovania. Ak v budúcnosti nájdeme dostatočné uplatnenie a vyhodnotíme jeho použitie ako opodstatnené, sme pripravený sa k tomuto frameworku vrátiť.

8.5.2 Overenie implementácie agenta *Zigorat*

- **Overenie komunikačnej vrstvy agenta *Zigorat*:** Komunikačná vrstva zabezpečovala odosielanie správ na server, prijímanie správ a obnovovanie informácií o okolitom svete. Overením na úrovni analýzy kódu, analýzy posiadaných/prijímaných správ a samotnej funkcionality robota môžeme konštatovať, že táto vrstva pracuje správne.
- **Overenie funkcionality na vyššej úrovni:** Podobne sme overili moduly správania pre pohyb jednotlivých kĺbov, na vyššej úrovni moduly chodenia a otáčania.

Funkcionalita vytvoreného prototypu nepokryla komunikačný protokol úplne. Z nášho pohľadu je však najdôležitejšia overenie funkcionality, ktorú predpokladáme pri implementácii evolučného algoritmu (založený na pohybe kĺbov).

8.5.3 Overenie nefunkcionálnych požiadaviek na server

Ako už bolo spomenuté (10.2.2 *Komunikácia robota*), evolučný algoritmus bude vyžadovať opakované pripájanie/odpájanie sa na/od server/serveru v krátkych časových intervaloch. Z využitím prototypu sme tiež overili schopnosť servera zvládnuť spomenuté zaťaženie.

Prototypovanie z pohľadu splnenia cieľov považujeme za úspešné. Za dôležité považujeme hlavne overenie funkcionality súvisiacej s implementáciou evolučného algoritmu a overenie architektonického návrhu agenta. Prototypovaním sme minimalizovali riziko výskytu problémov pri implementácii v letnom semestri.

9. Revízia č.1

Táto kapitola vznikla ako reakcia na posudok tímu č.11 odovzdaný dňa 22.11.2007 na dokumentáciu k projektu. Doplnená je kapitola 2.2 o simulačnom prostredí a kapitola 5 obsahujúca špecifikáciu požiadaviek.

9.1 Kapitola 2.2 Simulačné prostredie

Do tejto kapitoly sme sa na základe posudku rozhodli zaradiť podkapitolu o všeobecnom popise servera ako aj monitora ktorý slúži na vizualizáciu zápasu v simulovanom 3d futbale.

9.1.1 Soccer Server

Server zabezpečuje simuláciu robotického futbalu. Simuluje hru hráčov ovládaných nezávislými programami (agentmi). Architektúra systému agentov, server je typu klient-server. Agent v tejto architektúre reprezentuje klienta. Reálne sa hry zúčastňuje niekoľko agentov, kde každý ovláda jedného hráča hrajúceho za jeden z dvoch súperiacich tímov.

Server plní najmä nasledujúce úlohy:

- Uchováva aktuálny stav hry (skóre, čas, ...) a aktuálny stav jednotlivých hráčov.
- Prijíma informácie od agentov a na ich základe upravuje stav príslušných hráčov na ihrisku (agent ovláda efekty hráča).
- Simuluje pohyb hráčov a lopty na základe fyzikálnych zákonov a následne upravuje ich stav.
- Stará sa o dodržiavanie pravidiel a vyhodnocuje štandardné situácie na ihrisku.
- Informuje agentov o aktuálnom stave príslušných hráčov a stave samotnej hry (agent prijíma informácie z perceptorov hráča).
- Informácie o stave hry a stave jednotlivých hráčov posielajú monitorom (pozri kapitolu Monitor).

Na základe typu simulácie existuje niekoľko typov serverov robotického futbalu.

- 2D server: simulácia prebieha v 2D prostredí. Hráči sú reprezentovaný kruhmi s určitým polomerom.
- 3D server (Sphere): simulácia prebieha v 3D prostredí a hráči sú reprezentovaný guľami s určitým priemerom a hmotnosťou.
- 3D server (Humanoid): simulácia prebieha v 3D prostredí a hráči sú reprezentovaný humanoidmi. Ich telo pozostáva z niekoľkých častí navzájom prepojených kĺbmi.

V našom prípade sa venujeme simulácii s humanoidnými typmi hráčov a využívame preto príslušný server.

9.1.2 Monitor

Monitor predstavuje okrem agenta ďalší typ klienta komunikujúceho so serverom.

Monitor plní najmä nasledujúce úlohy:

- Prijíma zo servera informácie o stave hry a jednotlivých hráčov.
- Vizualne reprezentuje simuláciu na základe prijatých informácií.
- Umožňuje zaznamenať simuláciu do tzv. simulačného logu.
- Umožňuje prehrať zaznamenanú simuláciu v simulačnom logu.

Protokol využívaný na komunikáciu so serverom môže byť využitý aj na implementáciu trénera. Protokol monitora totiž umožňuje ovplyvniť stav hry a hráčov na ihrisku a tým iniciovať rôzne herné situácie.

9.2 Kapitola 5. Špecifikácia požiadaviek

Táto kapitola rozširuje pôvodnú kapitolu č.5 na základe posudku tímu č.11.

9.2.1 Schopnosti robota

Robot bude mať len základné pohybové schopnosti orientácie v priestore. Primárne sa bude jednať o schopnosť postaviť sa, prípadne udržanie rovnováhy. Tieto schopnosti však nebudú robotovi implementované priamo. Bude sa ich učiť prostredníctvom rôznych evolučných algoritmov. Zašpecifikujeme len počiatočnú polohu a metriky ktoré budú robotovi hovoriť že dosiahol svoj cieľ. Zvyšok už necháme na evolučnom vývoji.

9.2.2 Komunikácia robota

Robot bude pracovať už na novom type servera vo verzií 0.5.6. Komunikácia bude prebiehať prostredníctvom TCP protokolu. Počas evolúcie sa bude robot opakovane pripájať na server. Pri každom pripojení sa vykoná jedna generácia v evolučnom procese. Výsledné hodnoty sa zapíšu do súboru a robot sa odpojí. Následne sa proces bude opakovať. Robot sa pripojí, zo súboru si načíta predošlú generáciu, prebehne evolúcia a nové hodnoty sa opäť uložia do súboru.

9.2.3 Architektúra robota

Architektúru robota sme sa rozhodli prebrať z minuloročného projektu tímu 6th sense. Táto architektúra sa vyznačuje dobrou škálovateľnosťou a rozšíriteľnosťou. Použitý typ robota bol však určený pre starší typ servera kde bol výzor hráča značne zjednodušený (guľa). Z tohto dôvodu sme sa rozhodli komunikačnú časť t.j. časť ktorá komunikuje so serverom prebrať od humanoidného robota Zigorata. Tento robot je určený na nový typ servera a jeho komunikačná časť je dobre rozpracovaná. Ďalej by sme chceli do zdrojového kódu vniesť štandardy aby bol prehľadnejší a ľahko čitateľný.

9.2.4 Špecifikácia evolúcie

Evolučné algoritmy simulujú evolúciu, ktorá prebieha v prírode. Tento proces by mohol poskytnúť možnosť, ako nechať hráča aby sám získal základné schopnosti. Plánujeme otestovať možnosť využitia evolučných algoritmov. Budú využité na naučenie základných pohybov ako napríklad postavenie sa zo zeme, chôdza a pod. Ak sa evolučné algoritmy ukážu ako efektívny prístup, môžeme sa pokúsiť aplikovať ich aj na naučenie zložitejších pohybov. Použijeme kombináciu metód evolučného programovania a m-chromozómov. Z evolučného programovania využijeme reprezentáciu funkcie v tvare „koreňového stromu“. Ten bude reprezentovaný Readovým lineárnym kódom. M-chromozómy budú predstavovať jedincov premenlivej dĺžky.

10. Literatúra

- [1] RoboCup home page
<http://www.robocup.org/02.html> – oficialna stranka robocuou, (20.10.2007).
- [2] Socer Simulation
http://wiki.cc.gatech.edu/robocup/index.php/Soccer_Simulation, (21.10.2007).
- [3] Rules for 3D Soccer Simulation Competition during RoboCup 2007 Atlanta,
http://www.uni-koblenz.de/~murray/robocup/rc07/rules/3d_rules.pdf,
1. júl 2007.
- [4]. Hamid Reza Mohseni Nejad, Roya Helmmat Pour, Mohamad Mayanani, Amir Kharmandar, Hamid Bakhtiyari, Hojjat Nikan, Hamed Tirdad: Simulation of a Humanoid Soccer Robot, Team Description Proposal for RoboCup 2007, Sama 3D,
<http://www.uni-koblenz.de/~murray/robocup/rc07/Binaries/tdp/Sama3D.pdf>
- [5] Xu Yuan, Tan Yingzi: SEU-3D 2007 Soccer Simulation Team Description,
<http://www.uni-koblenz.de/~murray/robocup/rc07/Binaries/tdp/SEU-3D-TDP07.pdf>
- [6] Zigorat RoboCup teams,
<http://zigorat3d.googlepages.com/home2>
- [7] Xichao Xia, Haonam Ma: DNU_Explorer 3D Team Description 2007,
http://www.uni-koblenz.de/~murray/robocup/rc07/Binaries/tdp/DNU_Explorer_3D_TDP_2007.pdf
- [8] Ling Gao, Guangbin Cui, Pu Li, Hongxia Chai, Xiaomin Tang.: Fantasia 2007 Team Description,
<http://www.uni-koblenz.de/~murray/robocup/rc07/Binaries/tdp/fantasia2007tdp.pdf>
- [9] Yoichi Setoguchi, Nobuhiro Ito: The World Model for Autonomous Soccer Agents, NAITO-Strikers,
<http://www.uni-koblenz.de/~murray/robocup/rc07/Binaries/tdp/NAITO-StrikerS2007.pdf>
- [10] Saeid Akhavan, Mohammad Babaeizadeh, Hamid Reza Hasani, Arefeh Kazemi, Hoda Safaeipour : UI-AI3D Team Description,
<http://www.uni-koblenz.de/~murray/robocup/rc07/Binaries/tdp/uiai2007TDP.pdf>
- [11] S-expression,
<http://en.wikipedia.org/wiki/S-expression>, (20.10.2007)
- [12] Vladimír Kvasnička: Genetický algoritmus,
http://www2.fiit.stuba.sk/~kvasnicka/NeuralNetworks/5.prednaska/GA_background.pdf
- [13] V. Kvasnička, J. Pospíchal, P. Tiňo: Evolučné algorikmi, Slovenská technická univerzita v Bratislave, 2000, ISBN 80-227-1377-5
- [14] Obst, O., Rollmann - M., Spark, A Generic Simulator for Physical Multi-agent Simulations, 2005.

- [15] Smith, R.: Open Dynamics Engine v0.5 User Guide, 2006,
http://ode.org/ode-latest-userguide.html#sec_3_5_0, (14.11.2007).
- [16] Writing your own agent, Simspark
Wiki,http://simspark.sourceforge.net/wiki/index.php/Getting_Started#Linux
(14.11.2007).
- [17] The RoboCup Soccer Simulator, rcserver3D 0.5.6,
http://sourceforge.net/project/showfiles.php?group_id=24184 (14.11.2007).
- [18] The RoboCup Soccer Simulator, Email Archive: sserver-three-d,
http://sourceforge.net/mailarchive/forum.php?forum_name=sserver-three-d
(14.11.2007).
- [19] Little Green BATS: Installing Robocup 3D simulation server on Ubuntu,
<http://www.littlegreenbats.nl/?q=node/70> (12.12.2007).

11. Prílohy

11.1 Príloha A - Príklad dát prijatých zo servera

```

(time
(now 2884.39))
(GS
(t 0.00)
(pm BeforeKickOff))
(GYR
(n torso)
(rt 0.00 0.01 0.04))
(See
(F1L
(pol 35.14 126.87 -5.19))
(F2L
(pol 21.61 -169.21 -8.45))
(F1R
(pol 40.44 44.00 -4.51))
(F2R
(pol 29.45 -7.85 -6.19))
(G1L
(pol 26.94 141.85 -7.58))
(G2L
(pol 22.58 160.35 -9.05))
(G1R
(pol 33.55 29.64 -6.08))
(G2R
(pol 30.16 14.50 -6.76))
(B
(pol 13.04 71.57 -14.00))
(P
(team RoboLog)
(id 10)
(pol 10.50 90.01 0.00)))
(UJ
(n laj1_2)
(ax1 0.00)
(ax2 -0.00))
(UJ
(n raj1_2)
(ax1 -0.00)
(ax2 0.00))
(HJ
(n laj3)
(ax 0.00))
(HJ
(n raj3)
(ax 0.00))
(HJ
(n laj4)
(ax 0.00))
(HJ
(n raj4)
(ax -0.00))
(HJ
(n llj1)
(ax -0.00))
(HJ
(n rlj1)
(ax 0.00))

```

```
(UJ
  (n llj2_3)
  (ax1 0.00)
  (ax2 -0.00))
(UJ
  (n rlj2_3)
  (ax1 -0.00)
  (ax2 0.00))
(HJ
  (n llj4)
  (ax 0.00))
(HJ
  (n rlj4)
  (ax -0.00))
(FRP
  (n lf)
  (c 0.10 0.08 -0.05)
  (f -0.42 -0.30 12.22))
(UJ
  (n llj5_6)
  (ax1 0.00)
  (ax2 0.00))
(FRP
  (n rf)
  (c -0.10 0.08 -0.05)
  (f 0.41 -0.28 12.30))
(UJ
  (n rlj5_6)
  (ax1 -0.00)
  (ax2 -0.00))
```

11.2 Príloha B - Inštalácia servera

Postup by mal fungovať pre distribúcie *Dapper Drake (6.06)*, *Edgy Eft (6.10)* a *Feisty Fawn (7.04)*. Postup by mal byť podobný aj pri ostatných distribúciách ako *Fedora*, *Suse*, *Debian*.

- – príkazy.
- – komentáre k príkazom.

11.2.1 Povolenie *Universe* a *Multiverse* repozitárov

- `sudo gedit /etc/apt/sources.list`
- Podľa inštrukcií v súbore povoliť *Universe* a *Multiverse* repozitáre.
- `sudo apt-get update`

11.2.2 Inštalácia potrebných balíčkov

- `sudo apt-get install g++ ruby1.9 ruby1.9-dev libode0-dev libboost-dev libsdl-dev libfreetype6-dev libdevil-dev autoconf automake1.9 libtool freeglut3-dev tetex-extra cvs xlibs-dev libtiff4-dev libslang1-dev libboost-thread-dev libmng-dev libpng12-dev`
- V prípade problémov s *xlibs-dev* treba použiť *xlibs-static-dev*.

11.2.3 Vytvorenie liniek pre ruby

- `sudo rm /usr/bin/ruby`
- `sudo ln -s /usr/bin/ruby1.9 /usr/bin/ruby`
- `sudo ln -s /usr/lib/libruby1.9.so /usr/lib/libruby.so`

11.2.4 Stiahnutie zdrojových súborov servera z CVS repozitára

- Zdrojové súbory sa stiahnu do adresára `/home/<home_užívateľa>/rcsoccersim/`.
- `cvs -`
`d:pserver:anonymous@sserver.cvs.sourceforge.net:/cvsroot/sse`
`rver login`
- Prípadný prompt na heslo potvrdiť enterom.
- `cvs -z3 -`
`d:pserver:anonymous@sserver.cvs.sourceforge.net:/cvsroot/sse`
`rver co -P rcsoccersim/rcssserver3D`

11.2.5 Build a inštalácia servera

- `cd rcsoccersim/rcssserver3D/`
- `./bootstrap`
- `./configure`
- V prípade chyby súvisiacej s *ruby.h* treba odinštalovať *ruby* balíky, *ruby* stiahnuť a nainštalovať manuálne zo zdrojových súborov.
 - `sudo apt-get remove libruby*`
 - Stiahnuť zdrojové súbory *ruby* z <http://www.ruby-lang.org/en/downloads/>.
 - `sudo tar -xzf ruby-<stiahnutá_verzia>.tar.gz`
 - `cd ruby-<stiahnutá_verzia>/`
 - `./configure --enable-shared`
 - `make`

- o sudo make install
- o cd /usr/lib
- o sudo ln -s /usr/local/lib/libruby.so.1.8
- o sudo ldconfig
- o cd /home/<home_užívateľa>/rcsoccersim/rcssserver3D/
- o ./configure
- o make
- o sudo make install
- o sudo gedit /etc/ld.so.conf
- Do súboru pridať riadok '/usr/local/lib' ak sa v ňom ešte nenachádza.
- o sudo ldconfig

11.2.6 Inštalácia modelov a textúr

- o cd /usr/local/share/rcssserver3d
- o sudo wget http://downloads.sourceforge.net/sserver/rcssserver3d-0.5.6-data.tar.gz?use_mirror=mesh
- o sudo tar -xzf rcssserver3d-0.5.6-data.tar.gz

11.2.7 Spustenie servera

- o cd
- o simspark

11.2.8 Spustenie defaultného agenta

- o agentspark

11.3 Príloha C - Popis tried evolúcie

Trieda strom

Koreňový strom vyjadrujúci funkciu.

Premenné

- Vektor int (RLK) – Readov lineárny kód stromu
- Vektor structov – Vektor ohodnotení vrcholov stromu
 - Struct: union(id(int), integer, float), typ(int) – Typ označuje čím je vrchol ohodnotený. Podľa typu sa bude brať jedna z troch hodnôt unionu. Tieto hodnoty označujú identifikátor funkcie, identifikátor premennej a konštantu.

Konštruktory

- Konštruktor pre náhodný strom.

Metódy

- Vytvor náhodný strom – Pre potreby mutácie.
- Výmena podstromu, náhodného alebo daného – Mutácia a kríženie
- Výpočet svojej hodnoty
- Nájdi podstrom začínajúci na indexe x v RLK
- Vytvor a vráť kópiu podstromu zo seba ktorá začína na indexe x

Trieda gén

Gén chromozómu (jedinca).

Premenné

- Čas – Čas, kedy s má gén vykonať. Čas bude relatívny na začiatok vykonávania génu
- Id behavioru – Identifikátor správania.
- Vektor stromov – Parametre správania.

Konštruktory

- Konštruktor pre náhodný gén

Metódy

- Vykonanie génu
- Mutácia génu – Mutácia stromov a zmena času.
- Kríženie dvoch génov (výmena stromov alebo kríženie stromov)
- Nastav čas génu

Trieda chromozóm

Jedinec populácie.

Premenné

- Vektor génov – DNA informácia chromozómu
- Fitness – Ohodnotenie jedinca.

Konštruktory

- Konštruktor pre náhodný chromozóm
- Konštruktor s danými génmi – Pre potreby kríženia.
- Konštruktor zo súboru – Pre načítanie jedincov zo súboru.

Metódy

- Ulož do súboru
- Nastav fitness
- Vykonávanie (??? Zatiaľ nevyriešené)
- Vráť zoznam génov pre čas – Pre potreby vykonania jedinca.

Behavior evolúcia

Tento behavior bude zabezpečovať evolúciu.

Premenné

- Zoznam chromozómov – Populácia jedincov
- Stav – Stav behavioru. Označuje či sa vykonávajú jedinci, či sa vytvára nová generácia, ...
- Parametre evolúcie – pravdepodobnosti mutácií, veľkosť populácie, počet rodičovských párov, ...

Konštruktory

- Konštruktor pre vytvorenie behavioru s náhodnou populáciou
- Konštruktor pre vytvorenie behavioru s načítaním populácie zo súboru

Metódy

- Ulož populáciu do súboru
- Vykonaj jedinca (čas) – Vykoná jedinca. Gény jedinca majú relatívny čas, preto bude potrebný prepočet z absolútneho času.
- Ohodnot' jedinca – Vracia fitness jedinca.
- Reprodukcia (výber rodičov, kríženie a mutácia)
- Vytvorenie novej generácie

11.4 Príloha D – CD nosič

Na priloženom CD nosiči sa nachádza dokumentácia, ako aj zdrojové kódy serveru a hráča. CD má nasledujúcu štruktúru:

- \
- \Dokumentácia (obsahuje dokumentáciu k projektu)
- \Sources\player (obsahuje zdrojové kódy hráča neurotics agent)
- \Sources\server (obsahuje zdrojové kódy servera vo verzií 0.5.6)

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

RoboCup – tretí rozmer *(Riadenie projektu)*

Tím č. 17: Neurotics

Členovia tímu: Aleš Katona, Gabriel Braniša, Peter Čimo,
Tomáš Labuda, Juraj Šimon, Ján Kolesár

Študijný program: IS/SI

Ročník prvý, inžinierske štúdium

Email: team17.robocup@gmail.com

Web: <http://www2.dcs.elf.stuba.sk/TeamProject/2007/team17is-si/>

Vedúci projektu: Ing. Marián Lekavý

Ak. rok: 2007/2008

Obsah

Obsah.....	i
Zoznam obrázkov.....	ii
Zoznam tabuliek.....	ii
1. Úvod.....	1
1.1 Účel dokumentu.....	1
1.2 Štruktúra dokumentu.....	1
2. Ponuka.....	2
3. Plán projektu.....	3
3.1 Hrubý predbežný plán projektu.....	3
3.2 Zjemnený plán za obdobie od 12.10.2007 do 15.11.2007.....	4
3.3 Ganttova schéma.....	5
4. Rozdelenie úloh v tíme.....	6
4.1 Krátkodobé úlohy.....	6
4.2 Dlhodobé úlohy.....	6
4.3 Podrobnejší opis úloh.....	6
5. Zápisy zo stretnutí.....	8
6. Štandardy tímu.....	9
6.1 Kódové štandardy.....	9
6.1.1 Odrážkovanie.....	9
9.1.1 Pomenovanie.....	10
9.1.2 Súbory a adresáre.....	11
9.1.3 Ukážky.....	11
10. Použitá literatúra.....	13
11. Prílohy.....	14
11.1 Príloha A – Ponuka tímu.....	1
11.2 Príloha B – Zápisnice zo stretnutí tímu.....	1
11.3 Príloha C – Štandardy tímu.....	1
11.4 Príloha D.....	1
11.4.1 Úplný podrobný plán za zimný semester.....	1
11.4.2 Ganttova schéma.....	3
11.5 Príloha E - Plán projektu na letný semester.....	1
11.5.1 Podrobný plán na letný semester.....	1

Zoznam obrázkov

Obr. 1 Ganttova schéma hrubého plánu	3
Obr. 2 Ganttova schéma za obdobie 15.10.2007 až 17.12.2007	5

Zoznam tabuliek

Tab. 1 Hrubý predbežný plán na zimný semester	3
Tab. 2 Zjemnený plán za obdobie 15.10.2007 až 15.11.2007	4
Tab. 3 Krátkodobé úlohy v tíme.....	6
Tab. 4 Dlhodobé úlohy v tíme.....	6
Tab. 5 Zjemnený plán od 15.10.2007 do 13.12.2007.....	2
Tab. 6 Podrobný plán na letný semester	1

1. Úvod

1.1 Účel dokumentu

V tomto dokumente sa nachádzajú informácie ohľadne riadenia a komunikácie v našom tíme.

1.2 Štruktúra dokumentu

Dokument je rozdelený na niekoľko častí. V prvej časti sa nachádza štruktúra tohto dokumentu. V druhej časti sa nachádza naša ponuka k predmetu Tvorba softvérového systému v tíme. Tretia kapitola obsahuje náš pôvodný hrubý plán na zimný semester ako aj skutočný zjemnený plán. Súčasťou je aj Ganttova schéma. V tretej kapitole uvádzame rozdelenie úloh v tíme, tak z dlhodobého, ako aj z krátkodobého hľadiska. Štvrtá kapitola obsahuje všetky naše zápisnice od 12.10.2007 do 9.11.2007. V piatej kapitole sú zadané štandardy tímu ktoré bude dodržiavať počas celého trvania projektu. V šiestej časti spomíname použitú literatúru. V prílohe A sa potom spomína ponuka nášho tímu. V prílohe B sa nachádzajú zápisnice zo všetkých stretnutí tímov. V prílohe C uvádzame štandardy tímu, v prílohe D sa nachádza zjemnený plán za obdobie celého zimného semestra. Nakoniec v prílohe E je uvedený podrobný plán na letný semester.

2. Ponuka

V prílohe A sa nachádza ponuka nášho tímu pôvodne vytvorená pre projekt ALUMNI.

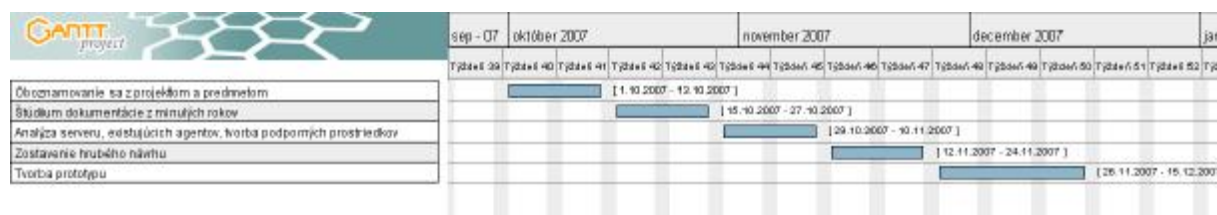
3. Plán projektu

Táto časť dokumentu obsahuje hrubý predbežný plán ktorý sme si stanovili na začiatku projektu. Pri tvorbe tohto dokumentu a po preštudovaní našich zápisníc sme zostavili skutočný plán podľa ktorého sme naozaj postupovali. Z tohto plánu sme zostavili Ganttovu schému.

3.1 Hrubý predbežný plán projektu

Obdobie	Popis	Stav
1.-2. týždeň	Oboznamovanie sa z projektom a predmetom	Ukončené
3.-4. týždeň	Štúdium dokumentácie z minulých rokov	Ukončené
5.-6. týždeň	Analýza serveru, existujúcich agentov, tvorba podporných prostriedkov	Ukončené
7.-8. týždeň	Zostavenie hrubého návrhu	Ukončené
9.-12. týždeň	Tvorba prototypu	Ukončené

Tab. 1 Hrubý predbežný plán na zimný semester



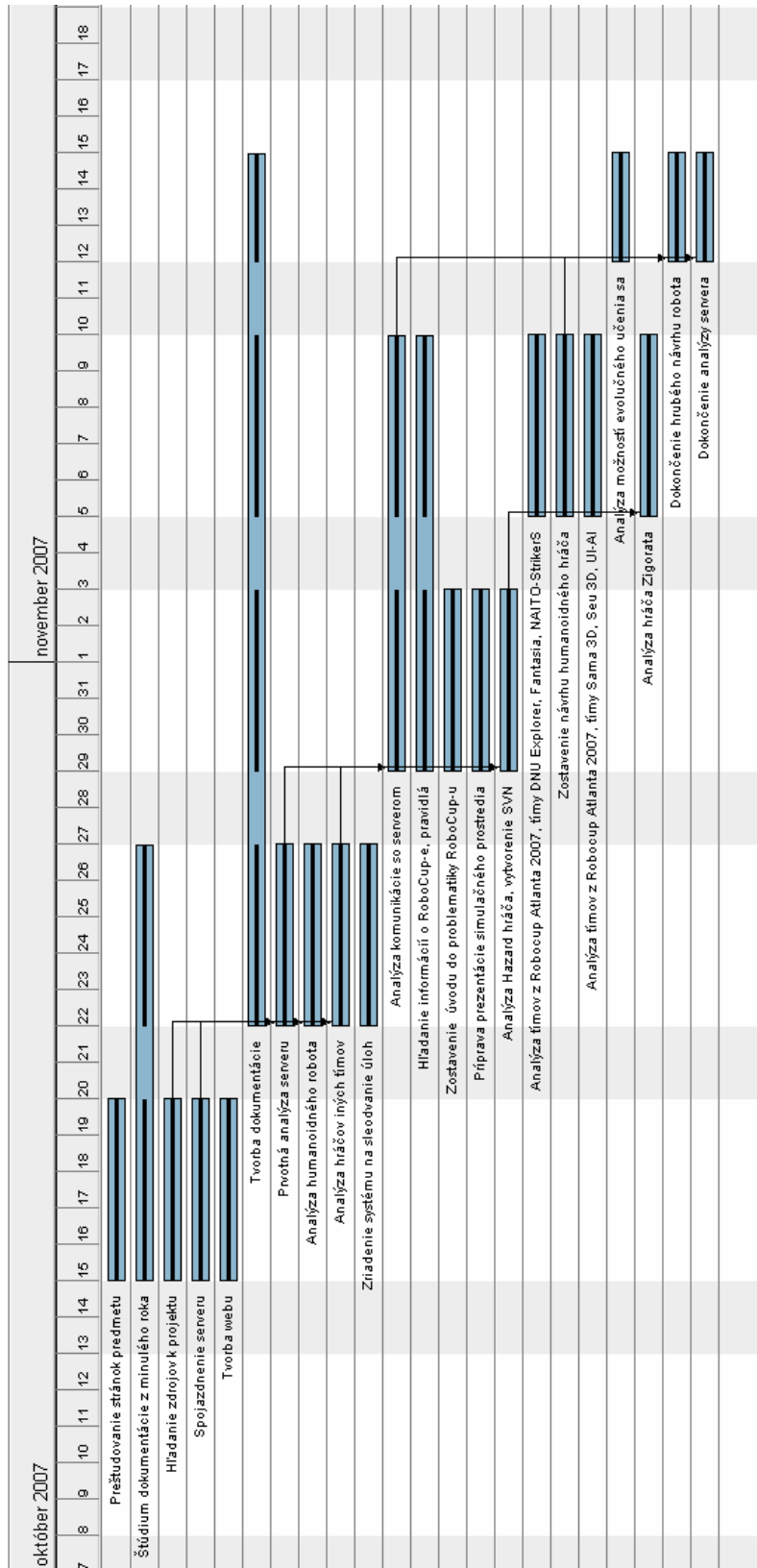
Obr. 1 Ganttova schéma hrubého plánu

3.2 Zjemnený plán za obdobie od 12.10.2007 do 15.11.2007

Úloha	Začiatok	Koniec	Zodpovedný
Preštudovanie stránok predmetu	15.10.2007	20.10.2007	Všetci
Štúdium dokumentácie z minulého roka	15.10.2007	27.10.2007	Všetci
Hľadanie zdrojov k projektu	15.10.2007	20.10.2007	Gabriel Braniša, Tomáš Labuda
Spojzdnenie serveru	15.10.2007	20.10.2007	Aleš Katona, Ján Kolesár
Tvorba webu	15.10.2007	20.10.2007	Peter Čimo, Juraj Šimon
Tvorba dokumentácie	22.10.2007	15.11.2007	Juraj Šimon
Prvotná analýza serveru	22.10.2007	27.10.2007	Tomáš Labuda
Analýza humanoidného robota	22.10.2007	27.10.2007	Ján Kolesár
Analýza hráčov iných tímov	22.10.2007	27.10.2007	Gabriel Braniša
Zriadenie systému na sledovanie úloh	22.10.2007	27.10.2007	Peter Čimo
Analýza komunikácie so serverom	29.10.2007	10.11.2007	Tomáš Labuda
Hľadanie informácií o RoboCup-e, pravidlá	29.10.2007	10.11.2007	Ján Kolesár
Zostavenie úvodu do problematiky RoboCup-u	29.10.2007	3.11.2007	Gabriel Braniša
Príprava prezentácie simulačného prostredia	29.10.2007	3.11.2007	Aleš Katona
Analýza Hazard hráča, vytvorenie SVN	29.10.2007	3.11.2007	Peter Čimo
Analýza tímov z Robocup Atlanta 2007, tímy DNU Explorer, Fantasia, NAITO-StrikerS	5.11.2007	10.11.2007	Peter Čimo
Zostavenie návrhu humanoidného hráča	5.11.2007	10.11.2007	Gabriel Braniša, Aleš Katona
Analýza tímov z Robocup Atlanta 2007, tímy Sama 3D, Seu 3D, UI-AI	5.11.2007	10.11.2007	Juraj Šimon
Analýza možností evolučného učenia sa	12.11.2007	15.11.2007	Ján Kolesár
Analýza hráča Zigorata	5.11.2007	10.11.2007	Gabriel Braniša, Aleš Katona
Dokončenie hrubého návrhu robota	12.11.2007	15.11.2007	Aleš Katona
Dokončenie analýzy servera	12.11.2007	15.11.2007	Tomáš Labuda

Tab. 2 Zjemnený plán za obdobie 15.10.2007 až 15.11.2007

3.3 Ganttova schéma



Obr. 2 Ganttova schéma za obdobie 15.10.2007 až 17.12.2007

4. Rozdelenie úloh v tíme

V našom projekte sme analyzovali nasledujúce krátkodobé a dlhodobé úlohy.

4.1 Krátkodobé úlohy

Činnosť	Zodpovedný
Analýza hráčov iných tímov	Peter Čimo, Juraj Šimon, Gabriel Braniša, Aleš Katona
Tvorba špecifikácie a návrhu	Aleš Katona
Analýza serveru a komunikácie	Tomáš Labuda
Inštalácia podporných webových prostriedkov	Peter Čimo
Inštalácia serveru, hráčov	Aleš Katona, Gabriel Braniša

Tab. 3 Krátkodobé úlohy v tíme

4.2 Dlhodobé úlohy

Meno	Úlohy
Juraj Šimon	tvorba dokumentácie a manažér vývoja, vedúci tímu
Peter Čimo	Manažér kvality, udržiavanie webu
Gabriel Braniša	Analytik, Tvorba plánu
Aleš Katona	Technické a podporné činnosti
Ján Kolesár	Manažér komunikácie
Tomáš Labuda	Systémový analytik

Tab. 4 Dlhodobé úlohy v tíme

4.3 Podrobnejší opis úloh

Vedúci tímu

- Komunikácia v tíme
- Pridelovanie úloh
- Kontrola plnenia úloh

Manažér kvality

- Dodržiavanie štandardov

Analytik, Systémový analytik

- Analýza údajov z externých zdrojov

Technické a podporné činnosti

- Systémové programovanie
- Podpora vývoja

Manažér komunikácie

- Komunikácia s ostatnými členmi tímu
- Synchronizácia medzi členmi tímu

Udržiavanie webu

- Manažment webového sídla
- Manažment SVN repozitára

Tvorba dokumentácie

- Vytváranie a spájanie častí dokumentácie do jedného celku

Manažér vývoja

- Konzultácia úloh s vedúcim projektu
- Návrh nových úloh
- Koordinácia úloh

5. Zápisy zo stretnutí

Príloha B obsahuje všetky zápisy z našich stretnutí od 12.10.2007 do 6.12.2007.

6. Štandardy tímu

Príloha C obsahuje štandardné šablóny dokumentov aké používame pri tvorbe zápisnice a šablónu preberacieho protokolu.

6.1 Kódové štandardy

Pri písaní zdrojového kódu, je pre jeho dobrú čitateľnosť, dobré dodržiavať nasledujúce zásady.

6.1.1 Odrážkovanie

Funkcie:

7. Typ návratovej hodnoty funkcie, názov funkcie a parametre funkcie v jednom riadku.
8. Otvárajúca a zatvárajúca zátvorka na novom riadku bez odsadenia.
9. Príklad:

```
int mojaFunkcie(int parameter)
{
    ...
}
```

Cykly, podmienky, ... :

- | Ak je telo na viac riadkov, otváracia a zatváracie zátvorky na nový riadok, samotné telo odsadiť (tabulátor).

Príklad:

```
if (parameter > 5)
{
    writeln(...);
    ...
}
```

- | Ak je telo jednoriadkove pokračovať v aktuálnom riadku bez nového riadku.

Príklad:

```
if (parameter = 9) writeln(...);
```

- | V cykloch používať názvy premenných *i*, *j*, *k*, *l*, ...

Príklad:

```
for (int i; i < 4; i++) for (int j; j < 4; j++) writeln(...);
```

- | Za príkazom *if*, *while*, ... nechať medzeru.
- | V podmienke pri zátvorkách nenechávať medzeru.

Príklad:

```
if (parameter = 9) ...
```

- | Pri aritmetických a logických operátoroch nechávať medzeru z oboch strán.

Príklad:

```
tmp += 5 + 6 * 53;
```

- | Kľúčové slovo *else* písať buď na nový riadok, pri jednoriadkovom *if* bez zátvoriek, alebo za ukončovaciu zátvorku, otváraciu zátvorku potom písať za *else* na tom istom riadku.

Príklad:

```
if (parameter > 5)
{
    writeln(...);
} else {
    writeln(...);
}
```

alebo:

```
if (parameter = 9) writeln(...);
else writeln(...);
```

Triedy, strucky:

- | Pri dedení, za názvom triedy, ktorá dedí bodkociarka bez medery a za nou mezdera.
- | Otvárajúca a zatvárajúca zátvorka na novom riadku bez odsadenia.
- | Kľúčové slova ako *private*, *protected*, *public* bez odsadenia.
- | Premenné a funkcie odsadiť (tabulator).
- | Najprv deklarovať premenné až potom funkcie.
- | Príklad:

```
class Moj aTri eda: publ ic MaterskaTri eda
{
private:
    int xxx;

public:
    int a;
    int b;

    Moj aTri eda();
    void vypisNi eco(char *text);
};
```

9.1.1 Pomenovanie

Premenné a funkcie:

- | Začíname malým písmenom, viacslovné premenné píšeme spolu, ďalšie slová píšeme s veľkým prvým písmenom.

Príklad: mojaFunkcia() , mojaPremenna

Triedy a strucky:

- I Začíname veľkým písmenom, viacslovné premenné píšeme spolu, ďalšie slová píšeme s veľkým prvým písmenom.
Príklad: MojaTrieda
- I Behavior triedy majú v názve ako posledné slovo „Behavior“
Príklad: BasicBehavior, RunToBallBehavior

Makrá:

- I Celý názov píšeme veľkými písmenami, slová v názve oddeľujeme podčiarkovníkmi
Príklad: MOJE_MAKRO

9.1.2 Súbory a adresáre

Pomenovanie súborov:

- Každá trieda je umiestnená v samostatnom .h (deklarácia) a .cpp (telá funkcií) súbore, názov súboru je odvodený z názvu triedy – celý názov malými písmenami, slová oddelené podčiarkovníkom.

Príklad:

trieda sa volá RunToBallBehavior

tak bude v súboroch run_to_ball_behavior.h a run_to_ball_behavior.cpp

Adresáre:

- Behavior moduly – adresár *behaviors/*.
- Ostatné moduly – adresár *others/*.

9.1.3 Ukážky

```
int mojaFunkcie(int parameter)
{
    int tmp;
    int viacSlovnaPremenna;

    if (parameter > 5)
    {
        writeln(...);
    } else {
        writeln(...);
    }

    while (parameter < 4)
    {
        tmp += 5 + 6 * 53;
    }

    if (parameter = 9) writeln(...);
    else writeln(...);
}
```

```
        for (int i;i < 4;i++) writeln(...);
    return 7;
}

struct Moj Struct
{
    int a;
    int b;
};

class Moj aTrieda: public MaterskaTrieda
{
private:
    int xxx;

public:
    int a;
    int b;

    Moj aTrieda();
    void vypisNi eco(char *text);
};

#define MOJE_MAKRO 1
```

10. Použitá literatúra

- [1] Bieliková M.: Tvorba systému v tíme I, II,
<http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html#dokumentacia>

11. Prílohy

11.1 Príloha A – Ponuka tímu

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

ALUMNI

Informačný systém pre komunikáciu s absolventmi

Ponuka

Odbor: Softvérové inžinierstvo
Dátum: 30.9.2007

Bc. Peter Čimo
Bc. Aleš Katona
Bc. Ján Kolesár
Bc. Tomáš Labuda
Bc. Milan Herda
Bc. Juraj Šimon

Zadanie

Naša FIIT má záujem vhodnou formou prezentovať verejnosti svojich absolventov. Zároveň by fakulta rada udržiavala s absolventmi neformálny odborný kontakt pomocou webu, a tiež by rada poskytla svojim absolventom rámec na neformálnu odbornú a spoločenskú komunikáciu medzi nimi navzájom. Zámerom projektu je vytvoriť informačný systém, ktorý bude plniť uvedené úlohy.

Ciele systému:

- I *Prezentovať základné informácie o absolventoch verejnosti.* Znamená to zabezpečiť vytvorenie a udržiavanie databázy absolventov a vhodne prezentovať základné informácie o jednotlivcovi verejnosti na webe bez obmedzenia prístupu. Predpokladáme stručnú informáciu o absolventovi, kedy študoval, absolvoval, jeho špecializácia, o téme jeho bakalárskej, prípadne diplomovej práce, prípadne abstrakt práce. Tu by bolo vítané rozšíriť prezentáciu o grafické vyjadrenie zamestnanosti a odbornosti absolventov z rôznych hľadísk, pokiaľ dokážeme od nich získať k tomu potrebné údaje.
- I *Sprostredkovať fakulte získavanie aktuálnych informácií o absolventoch v praxi.* Ide o aktualizáciu kontaktu, zamestnania, profesijného vývoja, odborného zamerania, oblasti, v ktorej je aktívny, sfér odborného záujmu a pod., tie, ktoré poskytne sám absolvent. Táto oblasť je veľmi citlivá, vyžaduje záujem o kontakt z oboch strán a je podmienená prísnou ochranou údajov s vhodne zorganizovaným autorizovaným prístupom. Je to dôležitá, pre fakultu užitočná úloha, treba ju uvažovať.
- I *Umožniť absolventom vzájomnú komunikáciu.* Má to byť jednoduchá a bezpečná komunikácia v informatickej komunite chránená starostlivo navrhnutými prístupovými právami pre skupiny autorizovaných účastníkov. Má slúžiť na neformálnu výmenu informácií v komunite kolegov, rovesníkov, odborníkov z praxe, ktoré komunite poskytne sám účastník. Okrem sprostredkovania kontaktu môže byť úlohou tejto časti systému informovať záujemcov o odborných aktivitách komunity, poskytnúť pre ne priestor – fórum, prípadne ďalšie vhodné činnosti.

Dôležité požiadavky:

- I zachovanie bezpečného prístupu k informáciám
- I jednoduché rozhranie
- I systém bez ďalších hardvérových nárokov
- I jednoduchá a bezpečná komunikácia medzi všetkými používateľmi systému
- I modularita a rozšíriteľnosť
- I škálovateľnosť
- I rozhranie pre získavanie štatistických dát pre fakultu
- I import základných údajov z fakultných IS
- I export do súboru, možnosť umiestniť vybrané údaje ako prezentáciu o absolventoch na pamäťové médiá
- I nasadenie do skúšobnej prevádzky v marci 2008

Projekt riešili vlnajšie tímy č.15 a 18, každému z nich sa podarilo vytvoriť vhodný základ systému. Viac informácií o ich postupoch a dosiahnutých výsledkoch riešenia nájdete na webových stránkach:

- I <http://www2.dcs.elf.stuba.sk/TeamProject/2006/team18/>
- I <http://www2.dcs.elf.stuba.sk/TeamProject/2006/team15/public.html/>

V tomto akademickom roku treba vytvorené systémy analyzovať, zhodnotiť stav, urobiť výber, rozšíriť vybraný systém o novú požadovanú funkčnosť a dotvoriť ho tak, aby ho tohoroční absolventi už mohli používať.

Predstavenie tímu

Bc. Aleš Katona:

Počas bakalárskeho štúdia nadobudol poznatky z jazykov C, C++, Java, C#, Object Pascal a PHP. Témou bakalárskeho projektu bola multiplatformová sieťová knižnica – Lnet. Od roku 2005 je živnostníkom v oblasti IT. Je členom open source tímu Free Pascal a prácu v menšom tíme pozná veľmi dobre. Na strednej škole absolvoval kurz Cisco CCNA, 4 semestre. Pri práci na komerčných projektoch získal praktické poznatky o databázach PostgreSQL, MySQL a jazyku PHP, web serveroch Apache a Lighttpd. Z osobnej iniciatívy získal praktické vedomosti o správe systému Linux.

Bc. Juraj Šimon

Počas bakalárskeho štúdia nadobudol základné poznatky z jazykov C,C++ a Java. Témou jeho bakalárskej práce bol Adaptívny hyper mediálny systém. Od konca roku 2005 je živnostníkom v oblasti IT. Z prác na projektoch získal praktické skúsenosti o programovaní v PHP, Visual Basicu a základy jazyka SQL.

BC. Ján Kolesár

Počas bakalárskeho štúdia získal skúsenosti s programovacími jazykmi C/C++, Java, Pascal, SQL a HTML. Pracoval v prostrediach MS Visual Studio, Eclipse, NetBeans, FireBird Maestro, Delphi a čiastočne v prostredí Lazarus. Skúsenosti s tvorbou a prácou s databázovými systémami získal na predmete Databázové systémy a momentálne ich rozvíja v komerčnom projekte s technológiami FireBird a Delphi.

Bc. Peter Čimo

Popri štúdiu sa už niekoľko rokov podieľa tvorbe robustných internetových portálov a internetových obchodov, kde sa snaží uplatňovať vedomosti z bakalárskeho štúdia (postupy charakteristické pre softvérové inžinierstvo). Okrem najrôznejších webových technológií ovláda aj programovacie jazyky C/C++ a Pascal a databázové prostredia ako MySQL, PostgreSQL či Oracle. Má skúsenosti s vývojom pod operačnými systémami Linux aj Windows.

Bc. Tomáš Labuda

Je absolventom bakalárskeho štúdia na FIIT STU v odbore Informatika. Téma jeho záverečného projektu bola „Modelovanie deployment prostredia pomocou UML2.0“. Vyše dvoch rokov pracuje v softvérovej firme, kde sa špecializuje na vývoj J2EE aplikácií. Má široké znalosti z vývoja webových aplikácií (hlavne na platforme Java, menej PHP a ASP.NET) a využívania webovo orientovaných technológií (Web Services, XML, XSLT, ...). Má rozsiahle znalosti jazyka UML, ktoré nadobudol hlavne pri práci na bakalárskom projekte.

Bc. Milan Herda**Motivácia k tomuto projektu**

Počas štúdia na vysokej škole človek nadobudne množstvo skúseností, získa nových priateľov, kolegov alebo spolupracovníkov z prostredia fakulty kde študuje, ako aj z prostredia celej univerzity. Po skončení štúdií a nástupu do zamestnania však mnohé z týchto kontaktov či už vedomky alebo nie zanikajú, strácajú predošlú pevnosť. Je to prirodzený proces, človek vystavený novému prostrediu naväzuje kontakty nové, súvisiace s jeho novým zameraním a cieľmi, a staré kontakty postupne upadajú do zabudnutia.

Je to však škoda, pretože z udržiavaných vzťahov medzi jednotlivými absolventmi navzájom ako aj medzi fakultou a jej absolventmi by mali nesporný prospech obe strany. Na jednej strane fakulta, ktorá by prostredníctvom udržiavaných vzťahov s absolventmi mala akúsi spätnú väzbu a vedela by lepšie zhodnotiť svoje úsilie. Vedela by kde sa jej študentom podarilo zamestnať, ako aj to či pracujú v odpore ktorý vyštudovali. Zároveň by tak mala akýsi archív všetkých absolventov. V neposlednom rade by sa však vedela pochváliť ich úspechmi a prilákať tak nových študentov do daného oboru. Na strane druhej máme absolventov ktorí by z udržiavaných kontaktov mohli získavať nové informácie, pomoc či radu, či dokonca aj naväzovať nové kontakty. Ale aby obe strany mohli ťažiť z takýchto spoločných vzťahov musí tu byť niečo, čo by poskytovalo dané služby. Musel by tu byť systém ktorý by zabezpečoval určitú komunikáciu medzi fakultou a jej absolventmi. V minulých rokoch bolo prostredníctvom tímových projektov na túto tematiku vypracovaných viacero projektov. Bol skonštruovaný základ takéhoto systému, ktorý však ešte nespĺňa všetky požiadavky nasadenia do praktickej prevádzky.

Je teraz na nás preštudovať tieto minulé projekty a to čo už bolo vykonané. Chceme pokračovať v tomto úsilí, a podporiť a vylepšiť tento projekt a pomôcť a popohnať ho na jeho ceste do praktického nasadenia.

Ponuka riešenia

V rámci návaznosti na minuloročné projekty by sme chceli prevziať pozitívne črty týchto projektov a využiť poznatky a skúsenosti členov nášho tímu z praxe. Jednou z hlavných črt by mala byť jednoduchosť v zmysle jednoduchého a intuitívneho ovládania, aby sa užívateľ nemusel s aplikáciou učiť pracovať, ale aby s ňou rovno pracoval. Projekt by bol vyhotovený ako moderná „web 2.0“ aplikácia poháňaná výkonným databázovým riešením v pozadí.

Predpokladané zdroje projektu**Vývojové prostriedky:**

Pre dokumentačné účely budú použité prostriedky na písanie dokumentov a grafov (MS Word, Open Office) a tvorbu diagramov (Dia).

Vývoj samotného riešenia bude prebiehať v heterogénnom prostredí. Na správu zdieľania sa použije verziovací systém SubVersion. Každý člen tímu si vyberie editor zdrojových kódov podľa vlastných preferencií.

Softvérové prostriedky:

Systém bude vyžadovať web server, avšak nie je nutné vybrať konkrétny web server. Výhodou riešenia je možnosť systém nainštalovať na rôznych serveroch bez väčších problémov. Jedinou požiadavkou na server je podpora PHP 5 ako modul, alebo cez CGI prípadne FastCGI.

Samotný systém bude implementovaný v jazyku PHP 5.

Pre správu dát použijeme databázový systém PostgreSQL, ktorý bol použitý predošlým tímom, a považujeme ho za vhodný.

Hardvérové prostriedky:

Procesor x86 - 32, alebo 64 bitový, 1 GHz

Pamäť - aspoň 512 MB

Úložný priestor - aspoň 2 GB

Ďalšie štyri témy ktoré nás zaujali

- | Tvorba testov s využitím LaTeXu
- | Distribuovaný systém na riešenie symetrickej hry
- | Analyzátor sociálnych sietí
- | Tvorba rozvrhov

Rozvrh členov tímu

		7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00
		7:50	8:50	9:50	10:50	11:50	12:50	13:50	14:50	15:50	16:50	17:50	18:50	19:50	20:50
Po	Čimo	APS		@NP				ML2		@TSST1		@VSS			
	Kolesár	APS					@NP			@TSST1		@VSS			
	Labuda	APS								@TSST1		@VSS			
	Šimon	APS		@NP				ML2		@TSST1		@VSS			
	Herda														
	Katona										@TIST1		@VIS		
Ut	Čimo									MSI		@MSI ¹		@MSI ¹	
	Kolesár									MSI		@MSI ¹		@MSI ¹	
	Labuda									MSI		@MSI ¹		@MSI ¹	
	Šimon									MSI		@MSI ¹		@MSI ¹	
	Herda														
	Katona											BMIS		@BMIS	
St	Čimo							@ML2							
	Kolesár	NS									@NS ²				
	Labuda	NS				@NS ²									
	Šimon							@ML2							
	Herda														
	Katona				PDT						FS				
Št	Čimo				NP			ASS							
	Kolesár				NP			ASS							
	Labuda				NP			ASS							
	Šimon				NP			ASS							
	Herda														
	Katona				VI			@VI			AIS				
Pi	Čimo														
	Kolesár														
	Labuda				@NP										
	Šimon														
	Herda														
	Katona							@FS							
Vysvetlivky															
	@cvičenie	prednáška													
	SI	IS													
1 (@MSI)	všetci skupina B – nedeterministické striedanie začiatkov o 17:00 a 19:00														
2 (@NS)	cvičenie = konzultácie => možnosť zúčastniť sa v ľubovoľnom termíne (9:00-17:00)														

Kontakt

Nasledujúca tabuľka obsahuje mailové kontakty na všetkých členov tímu.

Meno	Osobný kontakt	Školský kontakt
Aleš Katona	almindor@gmail.com	xkatonaa@is.stuba.sk
Juraj Šimon	sajmonphoenix@gmail.com	xsimonj@is.stuba.sk
Peter Čimo	gunxter@gmail.com	xcimo@is.stuba.sk
Tomáš Labuda	tomasko.labuda@gmail.com	xlabudat@is.stuba.sk
Ján Kolesár	jkolesar@gmail.com	xkolesarj@is.stuba.sk
Milan Herda		xherdam@is.stuba.sk

11.2 Príloha B – Zápisnice zo stretnutí tímu

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

Zápisnica zo stretnutia tímu

Číslo stretnutia: *1.*

Dátum stretnutia: *12.10.2007*

Čas: *neurčitý*

Miesto: *žiadne, pokyny zaslané prostredníctvom emailu*

Prítomní: -

Stretnutie viedol: -

Zápisnicu vypracoval: *Juraj Šimon*

Kontrola a hodnotenie úloh

Jednalo sa o naše prvé stretnutie a tak sme doteraz nemali zadelené žiadne úlohy k vypracovaniu.

Záznam zo stretnutia

Od Ing. Mariána Lekavého sme dostali email s inštrukciami o rozdelení úloh na nasledujúci týždeň.

Rozdelenie úloh

Znenie úlohy	Poverený
Preštudovanie stránok predmetu	všetci
Preštudovať dokumentáciu z minulého roka	všetci
Vyhľadanie odkazov na tematiku robotického futbalu	Tomáš Labuda, Gabriel Braniša
Rozbehať vlnajšie zdrojové kódy	Ján Kolesár, Aleš Katona
Vytvoriť webové sídlo tímu	Juraj Šimon, Peter Čimo

Zhrnutie

Dostali sme zadelenie úloh na najbližší týždeň, a odkaz na dokumentáciu minuloročného tímu. Stretnutie prebehlo úspešne.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Zápisnica zo stretnutia tímu

Číslo stretnutia: 2.
Dátum stretnutia: *19.10.2007*
Čas: *07:00*
Miesto: D202
Prítomní: *všetci*

Stretnutie viedol: Ing. Marián Lekavý
Zápisnicu vypracoval: *Juraj Šimon*

Kontrola a hodnotenie úloh

Z minulého stretnutia boli úlohy rozdelené nasledovne:

Znenie úlohy	Poverený
Preštudovanie stránok predmetu	všetci
Preštudovať dokumentáciu z minulého roka	všetci
Vyhľadanie odkazov na tematiku robotického futbalu	Tomáš Labuda, Gabriel Braniša
Rozbehať vlnajšie zdrojové kódy	Ján Kolesár, Aleš Katona
Vytvoriť webové sídlo tímu	Juraj Šimon, Peter Čimo

Zhrnutie vypracovania úloh

Všetci členovia tímu si dôkladne preštudovali stránky predmetu. Ďalej sme pozerali aj projekty z minulého aj predminulého roka. Tomáš Labuda a Gabriel Braniša vyhľadali na internete niekoľko odkazov na iné tímy robotického futbalu a stratégie vo futbale. Tieto odkazy boli umiestnené na našu stránku. Stránku vytvorili Juraj Šimon a Peter Čimo. Ďalej sme vyrobili náš plagát ktorý je taktiež umiestnený na stránke. Ján Kolesár a Aleš Katona sa pokúšali rozbehať server a no zatiaľ niektoré knižnice stále nejdú.

Záznam zo stretnutia

Spoločne sme sa stretli a prebrali náš doterajší postup. Ing. Lekavý nás podrobnejšie oboznámil s úlohou a so skúsenosťami s robotickým futbalom z minulých rokov. Zhodli sme sa na použití novej verzie serveru kde už figurujú humanoidný roboti. Všetky predošlé projekty boli vytvorené v staršej verzii. Naším cieľom je teda zistiť aké sú rozdiely medzi starou a novou verzou. Úloha rozbehania servera stále ostáva. Ďalšou úlohou bude nastudovať si základnú koncepciu hráča aký sa používa pri novom serveri. Z administratívnych úloh je to potom vytvorenie systému pre zapisovanie úloh a ich trvanie. Ďalej bol zvolený vedúci tímu Juraj Šimon ktorý bude vykonávať aj funkciu správcu dokumentácie. Za dodržiavanie štandardov v zdrojovom kóde bude zodpovedný Peter Čimo. Za dodržiavanie termínov bude zodpovedný Gabriel Braniša. Bol upravený čas stretnutí na 8:00 vo štvrtok.

Rozdelenie úloh

Znenie úlohy	Poverený
Príprava a začatie písania dokumentácie, požiadavky	Juraj Šimon
Preštudovať dokumentáciu z minulého roka	všetci
Pohľadať čo najviac info o novom serveri, dokumentácia, fóra, tipy triky, proste všetko	Tomáš Labuda
Vyhľadanie čo najviac informácií o fungovaní humanoidného robota v robotickom futbale	Ján Kolesár
Naplniť systém úloh termínmi, vyhľadať čo najviac humanoidných hráčov na nete info o nich	Gabriel Braniša
Rozbehať vlnajšie zdrojové kódy	Aleš Katona
Vytvoriť systém na písanie úloh členov tímu	Peter Čimo

Zhrnutie

Bližšie sme sa oboznámili s tematikou robotického futbalu, spresnili termíny stretnutí, administratívne záležitosti a funkcie členov v tíme. Dostali sme rozdelené úloh na ďalší týždeň.

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

Zápisnica zo stretnutia tímu

Číslo stretnutia: 3.

Dátum stretnutia: *26.10.2007*

Čas: *08:00*

Miesto: D202

Prítomní: *všetci*

Stretnutie viedol: Ing. Marián Lekavý

Zápisnicu vypracoval: *Juraj Šimon*

Kontrola a hodnotenie úloh

Z minulého stretnutia boli úlohy rozdelené nasledovne:

Znenie úlohy	Poverený
Príprava a začatie písania dokumentácie, požiadavky	Juraj Šimon
Preštudovať dokumentáciu z minulého roka	všetci
Pohľadať čo najviac info o novom serveri, dokumentácia, fóra, tipy triky, proste všetko	Tomáš Labuda
Vyhľadanie čo najviac informácií o fungovaní humanoidného robota v robotickom futbale	Ján Kolesár
Naplniť systém úloh termínmi, vyhľadať čo najviac humanoidných hráčov na internete info o nich	Gabriel Braniša
Rozbehať vlnajšie zdrojové kódy	Aleš Katona
Vytvoriť systém na písanie úloh členov tímu	Peter Čimo

Zhrnutie vypracovania úloh

Podarilo sa nám nájsť postup rozchodenia nového serveru. Server sme úspešne spustili. Ďalej sme našli niekoľko typov humanoidov a preštudovali ich schopnosti. Vytvorili sme systém na zápis úloh v podobe Gmail účtu. Naplnili sme toto konto termínmi odovzdaní, prednášok a stretnutí.

Záznam zo stretnutia

Ako obvykle na začiatku stretnutia sme prebrali náš doterajší postup. Ďalej sme sa zhodli na postupe po dobu prvého odovzdávania. Rozdelili sme si úlohy tak, aby sme naplnili tento prvý cieľ. Pokúsime sa rozanalyzovať niekoľko humanoidných hráčov, ich schopností a štruktúry. Ďalej budeme študovať systém komunikácie so serverom aká je štruktúra správ v komunikácií a podobne. Spolu s úvodom do problematiky robotického futbalu tieto informácie budú tvoriť analýzu problematiky. Ďalej preskúmame minuloročného hráča Hazarda a uvážime či bude z neho možné niektoré časti použiť. Napadla nás aj myšlienka evolučného učenia sa takže sa budeme venovať aj skúmaniu tejto problematiky. Nakoniec s pomocou týchto všetkých informácií zostavíme návrh riešenia resp. nového hráča.

Rozdelenie úloh

Znenie úlohy	Poverený
Zápisnica, zostavenie dokumentácie	Juraj Šimon
Komunikácia servera, štruktúra komunikácie, hľadanie nových hráčov	Tomáš Labuda
Prihlásenie sa do robocup fór, zistenie typov hier a ich pravidiel, zistenie možností učenia sa pomocou evolučných algoritmov.	Ján Kolesár
Zostavenie všeobecného úvodu do problematiky robotického futbalu, analýza humanoidného hráča	Gabriel Braniša
Analýza robotického tímu hráčov, nastavenie serveru a hráčov na prezentáciu jednoduchej hry	Aleš Katona
Analýza Hazard hráča a jeho použiteľnosti a štruktúry kódu. Vytvorenie SVN repozitára pre uloženie zdrojových kódov.	Peter Čimo

Zhrnutie

Prebrali sme náš doterajší postup a stanovili si cestu k termínu prvého odovzdania. Načrtli sme možnú stratégiu tvorby nového hráča v podobe učenia sa pomocou evolučných algoritmov. Pre zostavenia analýzy preskúame niekoľko humanoidných hráčov ako aj hráča z minulého roku. Na základe týchto informácií zostavíme potom návrh nášho riešenia humanoidného hráča.

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

Zápisnica zo stretnutia tímu

Číslo stretnutia: 4.

Dátum stretnutia: *05.11.2007*

Čas: *18:00*

Miesto: DE150

Prítomní: *všetci*

Stretnutie viedol: Juraj Šimon

Zápisnicu vypracoval: *Juraj Šimon*

Kontrola a hodnotenie úloh

Z minulého stretnutia boli úlohy rozdelené nasledovne:

Znenie úlohy	Poverený
Zápisnica, zostavovanie dokumentácie	Juraj Šimon
Komunikácia servera, štruktúra komunikácie, hľadanie nových hráčov	Tomáš Labuda
Prihlásenie sa do robocup fór, zistenie typov hier a ich pravidiel, zistenie možností učenia sa pomocou evolučných algoritmov.	Ján Kolesár
Zostavenie všeobecného úvodu do problematiky robotického futbalu, analýza humanoidného hráča	Gabriel Braniša
Analýza robotického tímu hráčov, nastavenie serveru a hráčov na prezentáciu jednoduchej hry	Aleš Katona
Analýza Hazard hráča a jeho použiteľnosti a štruktúry kódu. Vytvorenie SVN repozitára pre uloženie zdrojových kódov.	Peter Čimo

Zhrnutie vypracovania úloh

Začali sme pracovať na zostavení špecifikácie a dokumentácie. Tomáš Labuda spracováva komunikáciu so serverom, úloha je hotová na 50%. Zostáva ešte preložiť niekoľko odstavcov. Ján Kolesár zohnal pravidlá robocup3d z internetu z turnaja z roku 2007 v Atlante. Gabriel Braniša zostavil všeobecný úvod do robotického futbalu. Peter Čimo analyzoval kód hráča Hazarda z minulého roku a vytvoril SVN ktoré sme zatiaľ nenaplnili. Aleš Katona analyzoval hráčov The little green BATS. Rozbehnúť na serveri sa mu ich žiaľ zatiaľ nepodarilo. Juraj Šimon zostavil zápisnicu a začal postupne zostavovať dokumentáciu z dokumentov posielaných ostatnými členmi tímu.

Záznam zo stretnutia

Na začiatku sme si zhodnotili úlohy ktoré sme mali zadané a stav práce na nich. Každý vyjadril stupeň svojho postupu na úlohe. Bez dlhších rečí sme sa poradili o ďalšom postupe. Tomáš Labuda dokončí dokumentovanie serveru. Ján Kolesár bude pokračovať v štúdiu evolučných algoritmov aby tak mohol zostaviť časť návrhu nášho hráča. Peter Čimo a Juraj Šimon sa podrobnejšie pozrú na hráčov z robocup3d turnaja v Atlante z roku 2007. Aleš Katona spolu s Gabrielom Branišom sa pokúsia zostaviť prvotný hrubý návrh robota.

Rozdelenie úloh

Znenie úlohy	Poverený
Analýza tímov z robocup3d turnaja v Atlante: Sama3D, SEU-3D, UI-AI.	Juraj Šimon
Komunikácia servera, štruktúra komunikácie, hľadanie nových hráčov – dokončenie.	Tomáš Labuda
Možnosti evolučného učenia sa, určenie metrík pre meranie úspešnosti rôznych činností robota	Ján Kolesár
Zostavenie prvotného hrubého návrhu robota	Gabriel Braniša, Aleš Katona
Príprava prezentácie serveru a hráčov	Aleš Katona
Analýza tímov z robocup3d turnaja v Atlante: DNU Explorer, Fantasia, Naito Strikers	Peter Čimo

Zhrnutie

Prehodnotili sme náš doterajší postup. Zostavili sme časť dokumentácie obsahujúcej pravidlá robocupu a úvod do robocup prostredia. Pokračujeme v plnení zadaných úloh. Ešte zanalyzujeme niekoľko tímov z robocup3d turnaja v Atlante 2007. Určíme si metriky ktorými budeme merať úspešnosť agenta pri evolučnom učení sa. Pripravíme si prezentáciu reálnej práce servera a hráčov.

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

Zápisnica zo stretnutia tímu

Číslo stretnutia: 5.

Dátum stretnutia: 09.11.2007

Čas: 08:00

Miesto: D202

Prítomní: *všetci*

Stretnutie viedol: Ing. Marián Lekavý

Zápisnicu vypracoval: *Juraj Šimon*

Kontrola a hodnotenie úloh

Z minulého stretnutia boli úlohy rozdelené nasledovne:

Znenie úlohy	Poverený
Analýza tímov z robocup3d turnaja v Atlante: Sama3D, SEU-3D, UI-AI.	Juraj Šimon
Komunikácia servera, štruktúra komunikácie, hľadanie nových hráčov – dokončenie.	Tomáš Labuda
Možnosti evolučného učenia sa, určenie metrik pre meranie úspešnosti rôznych činností robota	Ján Kolesár
Zostavenie prvotného hrubého návrhu robota, naplnenie SVN	Gabriel Braniša, Aleš Katona
Príprava prezentácie serveru a hráčov	Aleš Katona
Analýza tímov z robocup3d turnaja v Atlante: DNU Explorer, Fantasia, Naito Strikers	Peter Čimo

Zhrnutie vypracovania úloh

Analýzu sme už skoro dokončili, zostáva dopracovať troch agentov z Atlanty a agenta Zigorata. Komunikáciu servera sme ešte nestihli plne dokončiť. Úspešne sme odprezentovali prostredie robocupu3d a vizualizačné prostredie. Taktiež analýza tímov Sama3D, SEU-3D a UI-AI je hotová a zakomponovaná do dokumentácie.

Záznam zo stretnutia

Na začiatku stretnutia sme oboznámili nášho vedúceho o postupe prác na našich úlohách. Ďalej sme diskutovali o návrhu. Zhodli sme sa že použijeme agenta Hazarda z minulého roku ako model správania. Avšak vzhľadom na to že sa odvtedy server zmenil ako model nižších činností ako je komunikácia so serverom alebo základné pohybové schopnosti z neho už zobrať nemôžeme. Podrobnejšie sme si rozobrali štruktúru agenta Zigorata a rozhodli sme sa že tento model základných činností preberieme od neho. Dohodli sme sa na termíne odovzdania dokumentácie nášmu vedúcemu. Pozreli sme si niekoľko inštruktážnych videí o evolučnom učení sa chodiť, humanoidného robota.

Rozdelenie úloh

Znenie úlohy	Poverený
Zostavenie dokumentácie k odovzdaniu, výmena dokumentácie s druhým tímom	Juraj Šimon
Komunikácia servera, štruktúra komunikácie, hľadanie nových hráčov – dokončenie.	Tomáš Labuda
Možnosti evolučného učenia sa, určenie metrík pre meranie úspešnosti rôznych činností robota	Ján Kolesár
Zostavenie prvotného hrubého návrhu robota	Aleš Katona
Analýza hráča Zigorata	Gabriel Braniša
Analýza tímov z robocup3d turnaja v Atlante: DNU Explorer, Fantasia, Naito Strikers	Peter Čimo

Zhrnutie

Prehodnotili sme náš doterajší postup. Prebrali sme náš hrubý návrh a dohodli sa na podrobnostiach. Rozdelili sme si úlohy tak aby spoločne viedli k úspešnému odovzdaniu dokumentácie.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Zápisnica zo stretnutia tímu

Číslo stretnutia: 6.

Dátum stretnutia: 15.11.2007

Čas: 08:00

Miesto: D202

Prítomní: *všetci*

Stretnutie viedol: Ing. Marián Lekavý

Zápisnicu vypracoval: *Juraj Šimon*

Kontrola a hodnotenie úloh

Z minulého stretnutia boli úlohy rozdelené nasledovne:

Znenie úlohy	Poverený
Zostavenie dokumentácie k odovzdaniu, výmena dokumentácie s druhým tímom	Juraj Šimon
Komunikácia servera, štruktúra komunikácie, hľadanie nových hráčov – dokončenie.	Tomáš Labuda
Možnosti evolučného učenia sa, určenie metrík pre meranie úspešnosti rôznych činností robota	Ján Kolesár
Zostavenie prvotného hrubého návrhu robota	Aleš Katona
Analýza hráča Zigorata	Gabriel Braniša
Analýza tímov z robocup3d turnaja v Atlante: DNU Explorer, Fantasia, Naito Strikers	Peter Čimo

Zhrnutie vypracovania úloh

Peter Čimo a Gabriel Braniša dokončili analýzu im pridelených hráčov. Aleš Katona v spolupráci s Gabrielom Branišom a Jánom Kolesárom zostavili hrubý návrh hráča. Hráč sa bude učiť svoje schopnosti za pomoci evolučných algoritmov. Možnosti evolučných algoritmov podrobne preskúmal Ján Kolesár pričom určil aj možnosti ohodnocovacích funkcií. Tomáš Labuda dokončil opis servera vo verzii 0.5.6. Juraj Šimon zostavil dokumentáciu k projektu. Dokumentácia bola odovzdaná ako vedúcemu nášho tímu Ing. Mariánovi Lekavému tak aj druhému tímu č.11.

Záznam zo stretnutia

Na začiatku stretnutia sme odovzdali vypracovanú dokumentáciu vedúcemu tímu Ing. Mariánovi Lekavému. Obe strany podpísali preberací protokol. Ďalej sme preberali možnosti reprezentácie génov pri evolučných algoritmoch aplikovaných na RoboCup. Následne sme si stanovili náš ďalší postup ktorým je tvorba prototypu. Rozdelili sme si úlohy medzi jednotlivých členov tímu.

Rozdelenie úloh

Znenie úlohy	Poverený
Overenie funkčnosti budúcej simulácie na serveri	Tomáš Labuda
Tvorba evolučných algoritmov pre robota	Ján Kolesár
Naplnenie SVN (server a hráč/hráči)	Gabriel Braniša
Tvorba posudku tímu č.11	Juraj Šimon, Aleš Katona
Určenie kódových štandardov	Peter Čimo
Nájdenie univerzálneho komunikačného VoIP nástroja	Aleš Katona

Zhrnutie

Odovzdanie dokumentácie prebehlo v poriadku. Dohodli sme sa na ďalšom postupe. Postupne budeme tvoriť prototyp. Začneme teoretickou tvorbou evolučných algoritmov a praktickým odskúšaním niektorých vlastností servera. Ďalej vytvoríme základný model hráča schopného komunikovať so serverom.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Zápisnica zo stretnutia tímu

Číslo stretnutia: 7.

Dátum stretnutia: 22.11.2007

Čas: 08:00

Miesto: Softvérové štúdio

Prítomní: *všetci*

Stretnutie viedol: Ing. Marián Lekavý

Zápisnicu vypracoval: *Juraj Šimon*

Kontrola a hodnotenie úloh

Z minulého stretnutia boli úlohy rozdelené nasledovne:

Znenie úlohy	Poverený
Overenie funkčnosti budúcej simulácie na serveri	Tomáš Labuda
Tvorba evolučných algoritmov pre robota	Ján Kolesár
Naplnenie SVN (server a hráč/hráči)	Gabriel Braniša
Tvorba posudku tímu č.11	Juraj Šimon, Aleš Katona
Určenie kódových štandardov	Peter Čimo
Nájdenie univerzálneho komunikačného VoIP nástroja	Aleš Katona

Zhrnutie vypracovania úloh

Tomáš Labuda overil že je možné na server pripájať a odpájať jedného alebo viacerých hráčov ľubovoľný počet krát. Poukázal aj na problém pri pripojení a vložení hráča na hraciu plochu kedy nie je možné vybrať mu špecifickú polohu ako napr. určité natočenie kĺbov. Gabriel Braniša do SVN vložil zdrojové kódy hráča a serveru. Juraj Šimon vypracoval posudok k projektovej dokumentácii tímu č. 11 a odovzdal ho vedúcemu projektu Ing. Mariánovi Lekavému ako aj tímu č. 11. Peter Čimo vypracoval návrh kódových štandardov. Aleš Katona odporučil ako komunikačný nástroj VoIP nástroj TeamSpeak. Ján Kolesár pokračuje v návrhu evolučných algoritmov.

Záznam zo stretnutia

Na stretnutí sme preberali hlavne návrh kódových štandardov. Ujasnili sme si niektoré nedorozumenia. Peter Čimo vypracuje o dohodnutých štandardoch dokument. Ďalej sme preberali implementáciu modulov do prototypu. Návod pre tvorbu modulov dostal za úlohu Aleš Katona. On následne rozdelí implementáciu všetkých modulov medzi jednotlivých členov tímu. Na konci stretnutia sme odovzdali posudok tímu č. 11 nášmu vedúcemu projektu..

Rozdelenie úloh

Znenie úlohy	Poverený
Implementácia modulov	Všetci
Návod na tvorbu modulu	Aleš Katona
Vypracovanie reakcie na posudok	Juraj Šimon

Zhrnutie

Vypracovali sme posudok dokumentácie tímu č. 11. Dohodli sme sa na kódových štandardoch a rozdelili si úlohy implementácie modulov do prototypu.

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

Zápisnica zo stretnutia tímu

Číslo stretnutia: 8.

Dátum stretnutia: 29.11.2007

Čas: 08:00

Miesto: Softvérové štúdio

Prítomní: *všetci*

Stretnutie viedol: Ing. Marián Lekavý

Zápisnicu vypracoval: *Juraj Šimon*

Kontrola a hodnotenie úloh

Z minulého stretnutia boli úlohy rozdelené nasledovne:

Znenie úlohy	Poverený
Implementácia modulov	Všetci
Návod na tvorbu modulu	Aleš Katona
Vypracovanie reakcie na posudok	Juraj Šimon

Zhrnutie vypracovania úloh

Aleš Katona a Peter Čimo spoločne začali implementovať moduly do prototypu. Juraj Šimon vypracoval reakciu na posudok tímu č. 11. Implementácia základných modulov z dôvodu nedostatku času ešte nebola ukončená. Ján Kolesár pokračoval v tvorbe evolučných algoritmov. Gabriel Braniša a Tomáš Labuda sa zaoberali podpornými činnosťami v tíme.

Záznam zo stretnutia

Z dôvodu nedostatku času bolo naše stretnutie rýchle. V stručnosti sme si zhrnuli náš doterajší postup. Prebrali sme implementáciu modulov a jej budúci postup. Dohodli sme sa na vyhodnení knižnice *zeitgeist* z implementácie prototypu.

Rozdelenie úloh

Znenie úlohy	Poverený
Doplnenie popisu servera do dokumentácie	Tomáš Labuda
Tvorba evolučných algoritmov pre robota	Ján Kolesár
Implementácia prototypu	Gabriel Braniša, Aleš Katona, Peter Čimo
Korekcia chýb v dokumentácií	Juraj Šimon

Zhrnutie

Zostavili sme reakciu na posudok tímu č.11. Implementácia modulov pomaly napreduje. Dokumentáciu je potrebné očistiť od chýb.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Zápisnica zo stretnutia tímu

Číslo stretnutia: 9.

Dátum stretnutia: 6.12.2007

Čas: 08:00

Miesto: Softvérové štúdio

Prítomní: *všetci*

Stretnutie viedol: Ing. Marián Lekavý

Zápisnicu vypracoval: *Juraj Šimon*

Kontrola a hodnotenie úloh

Z minulého stretnutia boli úlohy rozdelené nasledovne:

Znenie úlohy	Poverený
Doplnenie popisu servera do dokumentácie	Tomáš Labuda
Tvorba evolučných algoritmov pre robota	Ján Kolesár
Implementácia prototypu	Gabriel Braniša, Aleš Katona, Peter Čimo
Korekcia chýb v dokumentácií	Juraj Šimon

Zhrnutie vypracovania úloh

Tomáš Labuda vypracoval doplnenie popisu servera na základe posudku tímu č.11 ktoré bolo zaradené do dokumentácie. Juraj Šimon pridal do dokumentácie kapitolu revízie, v ktorej reagujeme na posudok tímu č. 11 a doplníme zmeny. Ján Kolesár pokračuje v tvorbe implementácií evolučných algoritmov. Zvyšok tímu postupne implementoval väčšiu časť prototypu.

Záznam zo stretnutia

Na začiatku stretnutia sme zhodnotili naplnenosť úloh z minulého týždňa. Preberali sme odovzdávanie projektovej dokumentácie. Dohodli sme približný dátum prezentácie nášho prototypu druhému tímu č.11. Ďalej sme rozoberali čo všetko treba doplniť do dokumentácie a ako sa tieto úlohy rozdelia medzi jednotlivých členov tímu.

Rozdelenie úloh

Znenie úlohy	Poverený
Návod na inštaláciu servera	Tomáš Labuda
Opis podrobného návrhu robota, doplnenie špecifikácie evolúcie	Ján Kolesár
Príprava prezentácie pre druhý tím	Gabriel Braniša
Implementácia prototypu, zostavenie dokumentácie prototypu	Peter Čimo, Aleš Katona
Doplnenie riadenia projektu	Juraj Šimon

Zhrnutie

Implementácia prototypu je takmer hotová. Implementácia evolučných algoritmov napreduje zatiaľ pomalším tempom. Sústreďíme sa na dokončenie prototypu a dopracovaní dokumentácie.

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

Zápisnica zo stretnutia tímu

Číslo stretnutia: 10.

Dátum stretnutia: 13.12.2007

Čas: 08:00

Miesto: Softvérové štúdio

Prítomní: *všetci*

Stretnutie viedol: Ing. Marián Lekavý

Zápisnicu vypracoval: *Juraj Šimon*

Kontrola a hodnotenie úloh

Z minulého stretnutia boli úlohy rozdelené nasledovne:

Znenie úlohy	Poverený
Návod na inštaláciu servera	Tomáš Labuda
Opis podrobného návrhu robota, doplnenie špecifikácie evolúcie	Ján Kolesár
Príprava prezentácie pre druhý tím	Gabriel Braniša
Implementácia prototypu, zostavenie dokumentácie prototypu	Peter Čimo, Aleš Katona
Doplnenie riadenia projektu	Juraj Šimon

Zhrnutie vypracovania úloh

Tomáš Labuda vypracoval návod na inštaláciu servera. Na podrobnom návrhu a popise prototypu ďalej pracujú Peter Čimo a Aleš Katona. Gabriel Braniša pripravuje prezentáciu pre druhý tím. Juraj Šimon pripravuje dokumentáciu na odovzdanie. Ján Kolesár doplnil špecifikáciu evolúcie.

Záznam zo stretnutia

Na stretnutie sme doniesli predbežnú verziu dokumentácie. Vedúci projektu nám dal cenné rady ako ju ešte vhodne doplniť.

Rozdelenie úloh

Znenie úlohy	Poverený
Doplnenie informácií k evolučným algoritmom	Ján Kolesár
Doplnenie informácií k prototypu	Peter Čimo, Aleš Katona
Príprava prezentácie pre druhý tím	Gabriel Braniša
Zhodnotenie prototypu	Tomáš Labuda
Doplnenie riadenia projektu	Juraj Šimon

Zhrnutie

Na stretnutí sme prebrali aké nedostatky ešte má naša dokumentácia a prebrali možnosti ich nápravy.

11.3 Príloha C – Štandardy tímu

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

Preberací protokol

Typ projektu: Tímový projekt I

Mená študentov: Juraj Šimon, Peter Čimo, Gabriel Braniša, Tomáš Labuda, Ján Kolesár, Aleš Katona

Študijný program: SI/IS

Názov práce: Robocup tretí rozmer

Predmet prebratia:

1.

Odovzdávajúci subjekt: Tím Neurotics

Preberajúci subjekt:.....

.....
Podpis zástupcu odovzdávajúcej strany

.....
Podpis zástupcu preberajúcej strany

V Bratislave dňa

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

Zápisnica zo stretnutia tímu

Číslo stretnutia: .

Dátum stretnutia:

Čas:

Miesto:

Prítomní:

Stretnutie viedol:

Zápisnicu vypracoval:

Kontrola a hodnotenie úloh

Z minulého stretnutia boli úlohy rozdelené nasledovne:

Znenie úlohy	Poverený
...	...

Zhrnutie vypracovania úloh

...

Záznam zo stretnutia

...

Rozdelenie úloh

Znenie úlohy	Poverený
...	...

Zhrnutie

...

11.4 Príloha D

11.4.1 Úplný podrobný plán za zimný semester

Úloha	Začiatok	Koniec	Zodpovedný
Preštudovanie stránok predmetu	15.10.2007	20.10.2007	Všetci
Štúdium dokumentácie z minulého roka	15.10.2007	27.10.2007	Všetci
Hľadanie zdrojov k projektu	15.10.2007	20.10.2007	Gabriel Braniša, Tomáš Labuda
Spojzdnenie serveru	15.10.2007	20.10.2007	Aleš Katona, Ján Kolesár
Tvorba webu	15.10.2007	20.10.2007	Peter Čimo, Juraj Šimon
Tvorba dokumentácie	22.10.2007	15.11.2007	Juraj Šimon
Prvotná analýza serveru	22.10.2007	27.10.2007	Tomáš Labuda
Analýza humanoidného robota	22.10.2007	27.10.2007	Ján Kolesár
Analýza hráčov iných tímov	22.10.2007	27.10.2007	Gabriel Braniša
Zriadenie systému na sledovanie úloh	22.10.2007	27.10.2007	Peter Čimo
Analýza komunikácie so serverom	29.10.2007	10.11.2007	Tomáš Labuda
Hľadanie informácií o RoboCup-e, pravidlá	29.10.2007	10.11.2007	Ján Kolesár
Zostavenie úvodu do problematiky RoboCup-u	29.10.2007	3.11.2007	Gabriel Braniša
Príprava prezentácie simulačného prostredia	29.10.2007	3.11.2007	Aleš Katona
Analýza Hazard hráča, vytvorenie SVN	29.10.2007	3.11.2007	Peter Čimo
Analýza tímov z Robocup Atlanta 2007, tímy DNU Explorer, Fantasia, NAITO-StrikerS	5.11.2007	10.11.2007	Peter Čimo
Zostavenie návrhu humanoidného hráča	5.11.2007	10.11.2007	Gabriel Braniša, Aleš Katona
Analýza tímov z Robocup Atlanta 2007, tímy Sama 3D, Seu 3D, UI-AI	5.11.2007	10.11.2007	Juraj Šimon
Analýza možností evolučného učenia sa	12.11.2007	15.11.2007	Ján Kolesár
Analýza hráča Zigorata	5.11.2007	10.11.2007	Gabriel Braniša, Aleš Katona
Dokončenie hrubého návrhu robota	12.11.2007	15.11.2007	Aleš Katona
Dokončenie analýzy servera	12.11.2007	15.11.2007	Tomáš Labuda
Overenie funkčnosti budúcej simulácie na serveri	19.11.2007	23.11.2007	Tomáš Labuda
Tvorba evolučných algoritmov pre robota	19.11.2007	7.12.2007	Ján Kolesár
Naplnenie SVN (server a	19.11.2007	23.11.2007	Gabriel Braniša

hráč/hráči)			
Tvorba posudku tímu č.11	19.11.2007	23.11.2007	Juraj Šimon, Aleš Katona
Určenie kódových štandardov	19.11.2007	23.11.2007	Peter Čimo
Nájdienie univerzálneho komunikačného VoIP nástroja	19.11.2007	23.11.2007	Aleš Katona
Implementácia modulov	24.11.2007	28.11.2007	všetci
Vypracovanie reakcie na posudok	24.11.2007	28.11.2007	Juraj Šimon
Doplnenie popisu servera do dokumentácie	30.11.2007	5.12.2007	Tomáš Labuda
Implementácia prototypu	30.11.2007	5.12.2007	Gabriel Braniša, Aleš Katona, Peter Čimo
Korekcia chýb v dokumentácií	30.11.2007	5.12.2007	Juraj Šimon
Návod na inštaláciu servera	6.12.2007	13.12.2007	Tomáš Labuda
Opis podrobného návrhu robota, doplnenie špecifikácie evolúcie	6.12.2007	13.12.2007	Ján Kolesár
Príprava prezentácie pre druhý tím	6.12.2007	13.12.2007	Gabriel Braniša
Opis podrobného návrhu robota	6.12.2007	13.12.2007	Aleš Katona
Implementácia prototypu, zostavenie dokumentácie prototypu	6.12.2007	13.12.2007	Peter Čimo
Doplnenie riadenia projektu	6.12.2007	13.12.2007	Juraj Šimon

Tab. 5 Zjemnený plán od 15.10.2007 do 13.12.2007

11.5 Príloha E - Plán projektu na letný semester

V tejto prílohe uvádzame podrobný plán projektu na letný semester.

11.5.1 Podrobný plán na letný semester

Obdobie	Popis	Podrobnosti
1. týždeň	Zhodnotenie prototypu a práce na zimnom semestri	<ul style="list-style-type: none"> Zhodnotenie práce zo zimného semestra Spresnenie plánu a činností v tíme Vytýčenie postupu a cieľov projektu v letnom semestri
2. týždeň	Zpracovanie zmien a pripomienok do prototypu a dokumentácie	<ul style="list-style-type: none"> Zpracovanie pripomienok z posudku prototypu a dokumentácie
3. – 4. týždeň	Implementácia tried evolúcie	<ul style="list-style-type: none"> Implementovanie objektov potrebných k evolúcií (gén, chromozóm, strom, populácia)
5. týždeň	Implementácia evolučného behavioru	<ul style="list-style-type: none"> Implementácia správania evolúcie do robota
6. týždeň	Testovanie a dolad'ovanie objektov evolúcie a evolučného behavioru	<ul style="list-style-type: none"> Zpracovanie objektov evolúcie do správania evolúcie Otestovanie interakcie správania evolúcie s objektmi evolúcie Dolad'ovanie chýb
7. – 8. týždeň	Spustenie evolúcie	<ul style="list-style-type: none"> Spustenie evolúcie s náhodnými hodnotami Sledovanie evolučného procesu
9. týždeň	Zhodnotenie výsledkov evolúcie	<ul style="list-style-type: none"> Vyhodnotenie výsledkov evolučného procesu
10. – 11. týždeň	Úprava evolučných algoritmov na základe výsledkov zhodnotenia a spustenie novej evolúcie	<ul style="list-style-type: none"> Zpracovanie výsledkov predošlej evolúcie Zmena konštánt, ohraničení príp. funkcií Spustenie nového evolučného procesu s upravenými hodnotami
12. týždeň	Vyhodnotenie výsledkov	<ul style="list-style-type: none"> Zhodnotenie výsledkov evolučného procesu ako aj výsledkov celej práce

Tab. 6 Podrobný plán na letný semester