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

ROBOCUP – tretí rozmer

(dokumentácia k 1. šprintu)

Tím č.17 : Tím 17 žije...

Bc. Filip Baďura
Bc. Roman Bilevic
Bc. Tomáš Blaho
Bc. Andrej Bisták
Bc. Peter Holák
Bc. Jozef Macho
Bc. Peter Paššák

Študijný program: Softvérové inžinierstvo/Informačné systémy
Ročník: 1.ročník inžinierskeho štúdia
Predmet: Tímový projekt
Vedúci projektu: Ing. Marián Lekavý, PhD.
Akademický rok: 2011/12

1. Príbeh(„user story“)

Požiadavky zadávateľa, resp. ciele, ktoré si tím určil pre daný šprint:

- Analýza tímu Androids
- Analýza vybraných tímov
- Analýza fyzikálneho modelu

2. Analýza

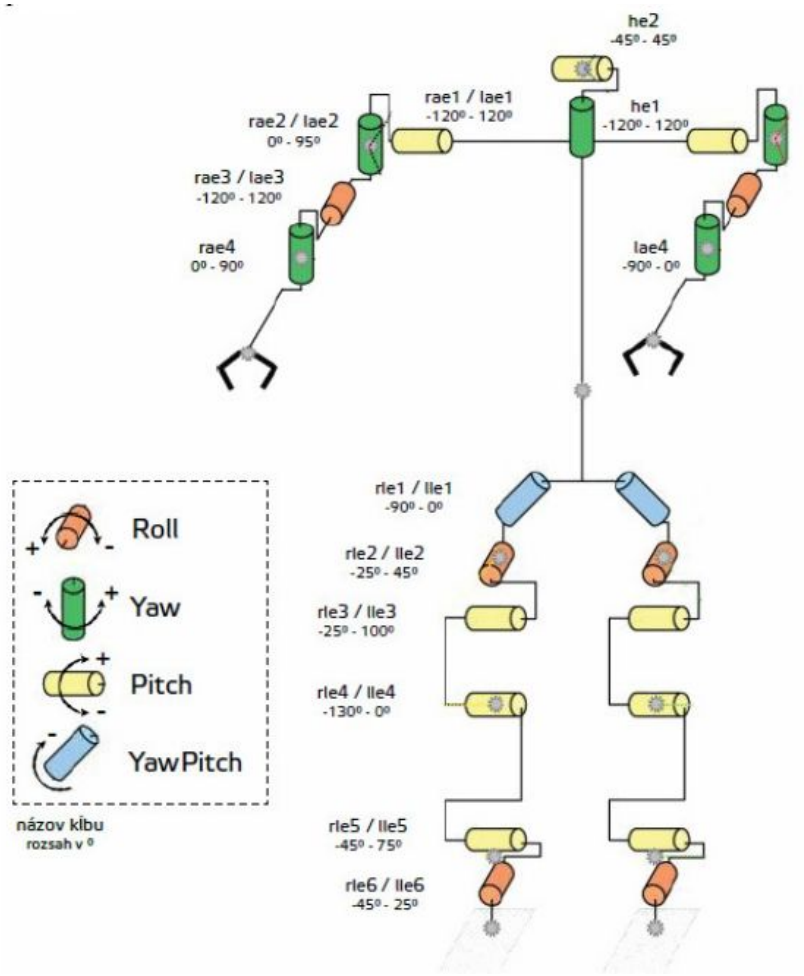
2.1. Androids

Tím Androids je zoskupenie študentov na predmete Tímový projekt na FIIT STU v školskom roku 2010/2011. V tejto časti analýzy sa sústreďujem na ich prácu, na ktorú by mal náš tým tento rok nadviazať. Analyzoval som ich konečnú dokumentáciu a používateľskú príručku, pomocou ktorej som následne úspešne vyskúšal ich konečný produkt.

Členovia tímu androdis v minulom roku analyzovali dvanásť svetových 3D tímov, server pre robocup a agenta JIM. Ďalej sa budeme stručne venovať iba robotovi NAOvi a agentovi JIMovi, pretože slúžil ako podklad pre vytvorenie agenta tímu androids.

2.1.1. Robot NAO

Model robota, s ktorého využili na projekte RoboCup 3D, je Robot NAO. Má výšku je 57 cm a váhu 4,5 kg. Je zložený z 22 kĺbov. Na Obr. 1 sú znázornené kĺby a osi robota NAO.



Obrázok č.1: Kĺby a osi robota NAO – pomôcka pri tvorbe pohybov

2.1.2. Agent JIM

Agent JIM je fakultný agent, pôvodne vytvorený z agenta Sirius tímu Critical Error a prenesený do Javy. Tím androids komunikáciou s autormi agenta získali dobrý prehľad o tomto agentovi. Rozhodli sa vylepšovať jeho koncept, pretože je vhodne štruktúrovaný a organizovaný, moderný a dostatočne zdokumentovaný.

Z ich analýzy sa dozvedáme, že agent bol vytvorený v prostredí Eclipse v jazyku Java, kde bola vytvorená komunikácia, parser, model sveta a predpovedací modul, a v jazyku Ruby, kde bola vytvorená logika a konfigurácie. Využité boli taktiež aspektovo-orientované techniky pomocou AspectJ, zmenili načítavanie pohybov a správania, a vytvorili pohyby vstávania z brucha, chrbta a prototyp chôdze. Agent sa skladá z piatich komponentov, a to Communication, Parser, Models, Planner a Movement Engine.

2.1.3. Špecifikácia a návrh prototypu

V špecifikácii sa venovali dosť širokému záberu požiadaviek. Chceli vylepšiť alebo vytvoriť nové pohyby hráča, ako vstávanie (3 typy), chôdza(2 typy), otáčanie(v rôznych uhloch a vojenské), kop do lopty(nastavenie sa na loptu a rôzne druhy kopov) a bránenie gólom(pád do boku a sadnutie s rozkročenými nohami), chceli vytvoriť vyššiu logiku(chôdza, vstaie, kop do lopty, vedenie lopty, zorientovanie sa, blokovanie strely), ktorá v agentovi JIMovi chýbala, chceli vylepšiť editor pohybov a správania, chceli definovať vyššiu logiku pomocou XABSL(prototyp funkcii a definície parametrov vyššej logiky) a chceli vytvoriť framework pre efektívne testovanie schopností hráča(automatizovanie a zrýchlenie simulácie a jej vyhodnocovania).

V návrhu už vynechali niektoré oblasti zo špecifikácie a určili si priority. Navrhli 2 typy vstávania, chôdzu(najvyššia priorita), otáčanie o 90°, 180° a vojenské otáčanie(vysoká priorita, pretože je rýchle, no náročne na stabilitu), kop do lopty(kop dopredu a stranou nohy), bránenie(oba spôsoby). V návrhu vyššej logiky spracovali uviedli diagramy a vstupné parametre: chôdza(cieľ), vstávanie(aktuálny stav), kop do lopty(cieľ a sila), vedenie lopty(cieľ), zorientovanie sa na zistenie svojej polohy a polohy lopty.

Navrhli rozšíriť existujúci editor pohybov a to zlepšením funkcionality a implementovaním editora správania(ako plugin). Zlepšenie funkcionality: navrhnuté pridanie nového panelu s možnosťou symetrického a zrkadlového ohýbania kĺbov alebo samostatne, refactoring kódu, doplnenie používateľského rozhrania a umožnenie zložitejších pohybov. V editore správania navrhli algoritmus práce produkčného systému: načítanie súboru s pravidlami, výber pravidlo z bázy poznatkov, pre zvolené pravidlo nájde kombinácie naviazania premenných a podmienkovej časti a na ne naviaž akciu na pravidlo, odfiltruj nesprávne a zaraď medzi akcie na vykonanie, ak neexituje aplikovateľná akcia tak ukonči proces, inak vykonaj prvú akciu a znova vyber pravidlo z bázy poznatkov. Navrhli taktiež vyššie správanie v XABSL, kde je opisované pomocou konečného stavového automatu, ktorý pozostáva z menších celkov, ktoré sú implementované v Jave. Navrhli tak zorientovanie sa a blokovanie strely. Ďalej navrhli novú architektúru testovacieho agenta, ktorý musí dokázať testovať schopnosti nižšej úrovne a taktiež vloženie viacerých schopnosti nižšej úrovne a testovať tak schopnosti vyššej úrovne. Testovací framework je zložený z netriviálnych blokov: parametre modelu(meniteľné parametre agenta/agentov), záznam zo servera(informácie o situácii), požadované konanie(zadáva užívateľ), vyhodnotenie konania(nižšia úroveň – pozície kĺbov, vyššia úroveň pozície agentov). Navrhli implementovať testovací framework v Jave za pomoci knižníc JGAP a JAGA, ktoré obsahujú funkcie na prácu s generickými algoritmami.

2.1.4. Prototyp

Do prototypu sa rozhodli implementovať tri časti: testovací framework, pohyby pre agenta JIM, a vylepšenie v editore pohybov tímu Agenty 007.

Testovací framework je aplikácia na posielanie príkazov agentom, ich plnenie a vyhodnocovanie na základe údajov získaných zo servera. Musí umožňovať testovanie nižších aj vyšších schopností hráča, konfiguráciu udalosti na ihrisku, testovanie viacerých hráčov a ich kooperácie, umožniť optimalizáciu hľadaním lepších parametrov. V aplikácii je nutná komunikácia agenta so serverom. V implementácii upravili agenta tak, aby prijímal príkazy z vonku. Bolo nutné definovať, čo má agent robiť a ako to vyhodnotiť. Na tu vybrali jazyk Ruby, kde budú uložené skripty v stanovenom tvare, ktoré potom bude agent spúšťať. Ďalej zistili, že je nutné prenášať súbory medzi frameworkom a agentom. Zvolili komunikáciu klient – server, keďže komunikácia má prebiehať aj medzi hráčmi spustenými na viacerých počítačoch. Implementovali open-source TFTP server na agenta a TFTP klient na strane frameworku. Server nastavili aby pri priatí súboru s menom “ruby.exec” bol tento súbor automaticky vykonaný ako Ruby skript.

Používateľ frameworku zadáva Ruby skript, v ktorom je definované čo sa má vykonať, časovo závislý test, ktorý určuje, či výsledný test môže byť pozitívny(časová efektivita, napr. nahrávka so zlým smerom nebude nikdy úspešná), test vypovedajúci, či výsledok akcie bol úspešný.

Vyhodnotenie konania spracuje vstupy od používateľa a servera a následne odosiela agentom súbory (XML alebo Ruby skripty so zmenenými parametrami) a skript v Ruby, ktorý sa okamžite vykoná na strane agenta. Zmena súborov má potenciál zaviesť do frameworku princípy umelej inteligencie.

Server je schopný logovať priebeh simulácie pomocou S-výrazov(obyčajné reťazce alebo ďalšie s výrazy). Najskôr sa zapíšu informácie o prostredí, potom informácie o stave hry a potom prebieha periodické zapisovanie čiastočných informácii o stave hry.

Informácia o prostredí má nasledujúcu štruktúru:

((<EnvironmentInformation>)(<SceneGraphHeader>)(<SceneGraph>)),

Informácia o stave hry má podobnú štruktúru ako tá o prostredí a vyzerá nasledovne:

((<GameState>)(<SceneGraphHeader>)(<SceneGraph>)).

Hlavička grafu scény obsahuje informácie o kompletnosti grafu scény a jej verzie. Jej štruktúra vyzerá nasledovne: (Name Version Subversion)

Príklady a vysvetlenia:

(EnvironmentInformation) :

((FieldLength 18)(FieldWidth 12)(FieldHeight 40)
(GoalWidth 2.1)(GoalDepth 0.6)(GoalHeight 0.8)
(FreeKickDistance 1.3)(WaitBeforeKickOff 2)
(AgentRadius 0.4)(BallRadius 0.042)(BallMass 0.026)
(RuleGoalPauseTime 3)(RuleKickInPauseTime 1)(RuleHalfTime 300)
(play_modes 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)
)

(<GameState>) : ((time 0)(half 1)(score_left 0)(score_right 0)(play_mode 0))

(Name Version Subversion): (RSG 0 1)

Name : RSG(plný popis prostredia) alebo RDS(iba inf. o zmenách)

Version: hlavná verzia grafu scény, vždy 0
Subversion: vedľajšia verzia grafu scény, vždy 1

(SceneGraph) - predstavuje logickú a priestorovú reprezentáciu scény. Je štruktúrovaný do stromu s koreňom na pozícií <0,0,0> bez rotácie. Každý vrchol v strome obsahuje jedného alebo viac nasledovníkov. Pozícia a rotácia každého nasledovníka je násobkom všetkých transformačných matíc z koreňa až k nasledovníkovi. Každý uzol, ktorý nie je zároveň aj listom je označený skratkou nd. Existuje viacero typov uzlov: základný(napr. (nd BN <contents>)), transformačný(4x4matica napr. (nd TRF (SLT nx ny nz 0 ox oy oz 0 ax ay az 0 Px Py Pz 1))), geometrický, ktorý opisuje tvar objektu(zložité, viacero typov).

Parsovanie správ sa skladá z monitorovania a samotného parsovania. Monitorovanie sa stará o príjem informácii počas behu simulácie na porte 3200 vo forme S-výrazov. Parosvacia časť spracuje informácie a vytvorí z nich objektový model, ktorý bude spracovaný časťou zodpovednou za reprezentáciu simulácie.

Skladá sa zo 4 balíčkov:
1. sk.fiit.testframework.parsing – obsahuje triedy zodpovedné za samotné parsovanie informácií, najdôležitejšie,
2. sk.fiit.testframework.parsing.models – obsahuje triedy, ktoré tvoria objektový model Informácií,
3. sk.fiit.testframework.parsing.models.messages – obsahuje triedy, ktoré reprezentujú prijaté správy zo servera,
4. sk.fiit.testframework.parsing.models.nodes – obsahuje triedy, ktoré reprezentujú uzly v grafe scény.

2.1.5. Vytvorené pohyby

Vstávanie z brucha - najskôr hráč uvedie všetky kĺby do neutrálnej polohy, okrem ramenných, ktoré pripažíí, následne rozkročí a úplne skrčí nohy, v ďalšej fáze postupne prinožuje a vystiera nohy, čím zároveň vstáva,v koncovej fáze vystierania hráč predpaží ruky, ktoré až do tejto fázy boli pripažené.

Vstávanie z chrbta - vystretie tela a pripaženie rúk, rýchle predpaženie a čiastočné rozkročenie, vďaka čomu sa hráč čiastočne posadí, dokončenie rozkročovania a pokrčenia nôh, a zároveň opätovné pripaženie, čím sa hráč, prevráti na brucho a skončí v polohe, ktorú opisuje druhá odrážka v opise vstávania z brucha, následne pohyb prebieha ako vstávanie z brucha.

Pád vpred a vzad – pomocné pohyby, hlavne na testovanie, jednoduchá a rýchla zmena uhla kĺbu členku, ktorý je zodpovedný za napnutie a pritiahnutie nártu.

Drobná pomalá chôdza dopredu – jeden krok je zložený z dvoch fáz, najskôr hráč prenesie váhu na opornú nohu a druhú nohu zdvihne a mierne vystrie v kolene, čím ju predsunie pred opornú nohu, zároveň rukou na strane opornej nohy pohne v ramene dopredu, nakoniec dostúpi na predsunutú nohu a prenesie na ňu váhu kvôli ďalšiemu kroku. Dĺžka kroku je približne pätina dĺžky hráčovho chodidla. Táto chôdza je veľmi pomalá a jej účelom je predovšetkým presné priblíženie k lopte bez jej odkopnutia.

Drobná pomalá chôdza dozadu - mierne odrazenie sa z päty pri predkopnutí nohy, lebo päta sa jemne zachytí o zem, nasleduje mierne zanoženie pri vrátení predkopnutej nohy späť na zem. Dĺžka kroku je približne tretina chodidla. Hráč výrazne predkopáva nohy a mierne poskakuje. Drobčivá rýchla chôdza dopredu - podobá sa drobnej chôdzi dozadu, hráč pri nej mierne nadskakuje. Táto chôdza má rovnaké fázy ako predošlé dve chôdze, odlišuje sa len v hodnotách otočení jednotlivých kĺbov a je pri nej výraznejšie prenášanie váhy na opornú nohu pri kroku. Je určená hlavne na prekonávanie väčších vzdialeností.

Otočenie o 90° - jediné implementované otočenie, aj to úspešné len na 50%, pracuje len s kĺbmi nôh, funguje do oboch strán, fázy: mierne pokrčenie nôh a rúk v inicializačnej fáze pre zvýšenie stability priblížením ťažiska k zemi, prenesenie váhy na opornú nohu a otočenie druhej nohy v bedrovom kĺbe úplne na stranu, prenesenie váhy na otočenú nohu a prinoženie pôvodnej opornej nohy, uvedenie pohybovaných kĺbov do pozície, v ktorej boli na konci inicializačnej fázy pred ďalším pokračovaní.

Priamy kop - je jeden z viacerých druhov pohybov, ktoré sme sa rozhodli pre hráča JIM vytvoriť. Ide o kop špicou chodidla pri zapojení bedrového, kolenného kĺbu a členku. Ide o kop, ktorý poslúži na ďaleké prihrávky a kopy na bránu. Viď dokumentáciu androids.

Kop do lopty hranou chodidla - v tomto prípade ide o kop do lopty hranou chodidla agenta. Nejde však o kop dopredu, ale do boku, tento kop umožní agentovi nahrávku do strany bez nutnosti zdĺhavo sa nastavovať k lopte, vzhľadom na možnosti a anatómiu agenta nejde o silný kop – na kop možno využiť iba jediný kĺb. Viď dokumentáciu androids.

Pád brankára na bok - je jeden z možných spôsobov zabránenia gólu, ktorý je vhodné použiť, ak brankár nestojí v ceste strele na bránu, ale je v bode, z ktorého pri páde na bok pretne svojím telom trajektóriu strely. Pohyb je implementovaný pre ľavú aj pravú stranu. Viď dokumentáciu androids.

Prevrátenie na chrbát - Je to pomocný pohyb pre uvedenie brankára do polohy, z ktorej dokáže vstať po tom, čo použil pohyb pád na bok, je implementovaný pre použitie z ľavého aj pravého boku. Pohyb pozostáva z dvoch fáz: v prvej hráč presunie nezaťaženú ruku a nohu za telo a zaťaženú ruku smerom pred telo, čo spôsobí, že sa prevráti na chrbát v druhej fáze pripaží a vráti bedrové kĺby do neutrálneho uhla.

2.1.6. Úpravy v editore pohybov

Vytvorili dve úpravy a to úpravu XML exportu pohybov a symetriu pohybov. Prvá menovaná úprava bola nutná, pretože export bol určený pre hráča Sirius, a teda museli vytvoriť export pre hráča JIM. Upravili pôvodnú triedu Xml exportu – XMLExporter. Editor pohybov je napísaný v jazyku C# a na generovanie XML je použitá knižnica System.Xml. Na dosiahnutie nového formátu boli použité metódy WriteStartElement(),WriteEndElement() a WriteAttributeString() triedy XmlTextWriter. Kompletný formát je zachytený pomocou XSD schémy, ktorá sa nachádza v adresári moves hráča JIM(Hlavička, Constants, Low_skills, Phases). V editore pohybov vytvorili políčko, v ktorom volíme, či chceme, aby sa symetrické kĺby pohybovali spoločne (nezávisle, symetrické, zrkadlové). Všetky úpravy ohľadom symetrie kĺbov boli robené v triedach balíka MotionEditor. Ďalej sa pokúsili o implementáciu previazania kĺbov. Nešlo o previazanie symetrických kĺbov, ale napríklad synchronizáciu bedrového kĺbu s kolenným. Narazili však na problémy: editor to nepodporuje, treba implementovať a odladiť zložité výpočty, vizualizácia sa správala neprirodzene pri prehraní pohybu.

2.2. Robocanes

Robocanes je americký tím z univerzity v Miami. V roku 2011 vyhrali RoboCup German Open, z desiatich zápasov 8 vyhrali a 2 remizovali, pričom inkasovali iba jeden gól a dali 33. Veľmi zaujímavý je program, ktorý používajú. Jeho názov je RoboViz a autorom je člen tohto tímu Justin Stoecker.

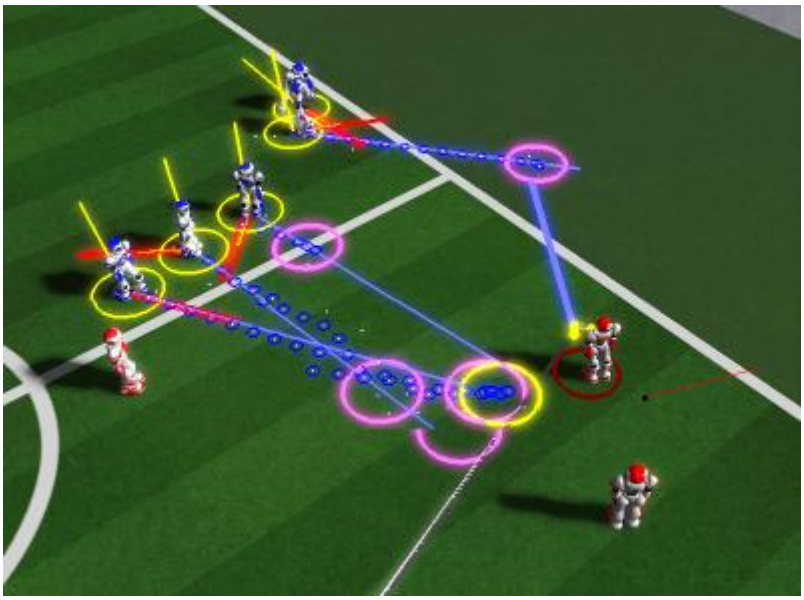
2.2.1. Roboviz

Roboviz je softvérový program vytvorený na hodnotenie správania vyvinutých agentov v multiagentovom systéme RoboCup 3D simulovaný robotický futbal. Je to interaktívne prostredie, v ktorom sú hráči a informácie o prostredí v 3D zobrazení. Roboviz do toho všetkého dopĺňa ďalšiu funkcionalitu ako zobrazovanie správania agentov a možnosť zasahovať do hry. Na obrázku nižšie je možné vidieť porovnanie medzi zobrazením v RoboViz vľavo a v SimSpark vpravo.

Obrázok č.2: RoboViz vs. SimSpark

2.2.2. Vizualizácia a ladenie

Roboviz umožňuje silné ladenie v reálnom čase so zobrazením správania robota pomocou geometrických útvarov. Toto umožňuje vývojárovi oveľa efektívnejšie analyzovať správanie a umožní lepšie ladenie algoritmov.



Obrázok č.3: Vizualizácia správania agentov

2.2.3. Hlavné používateľské rozhranie

Grafické rozhranie programu RoboViz je štandardné 3D zobrazenie s niekoľkými pridanými prvkami. Na vrchnej časti obrazovky je polopriehľadný panel obsahujúci informácie o stave hry. V strede je časomer a informácia o polčase a po oboch stranách sú mená tímov s príslušným skóre. V ľavo dole je zmenšený 2D obrázok ihriska, na ktorom sú hráči znázornení ako červené a modré bodky. Táto vymoženosť nie je zapnutá automaticky, ale dá sa spustiť stlačením klávesy „f“.

Čísla hráčov môžu byť zobrazované po zapnutí funkcie stlačením klávesy „i“. Zobrazujú sa nad hlavami hráčov. Veľa informácií je obsiahnutých priamo vo vnútri RoboViz okna, ako napríklad tabuľka so zoznamom možných grafických prvkov, ktorá sa dá premiestňovať a zväčšovať, alebo zavrieť.

Obrázok č.4: Hlavné okno

2.2.4. Zobrazenie a skrytie grafických prvkov

V programe RoboViz môžu byť znázornené rôzne jednoduché útvary, ktoré znázorňujú správanie robota. Sú to rôznofarebné čiary, kruhy lebo body rôznej veľkosti. V základnom nastavení sú zapnuté všetky vykreslenia, ale často je užitočné niektoré z nich vypnúť a sústrediť sa len na časť z nich. Panel sa zapína stlačením klávesy „p“. Tieto znázornenia sú mapované podľa mena.

2.2.5. Výber módu hry

V programe RoboViz sú na výber rôzne herné situácie. Názov aktuálnej hernej situácie je napísaný v hornom menu pod časomierou. Stlačením klávesy „o“ sa zobrazí menu, kde je možné zmeniť hernú situáciu. Na výber sú možnosti ako rozohrávka, rohový kop, penalta, priamy kop a mnohé ďalšie. Na túto funkcionalitu je potrebný rcssserver3d.

Obrázok č.5: Herné situácie

2.2.6. Presun objektov

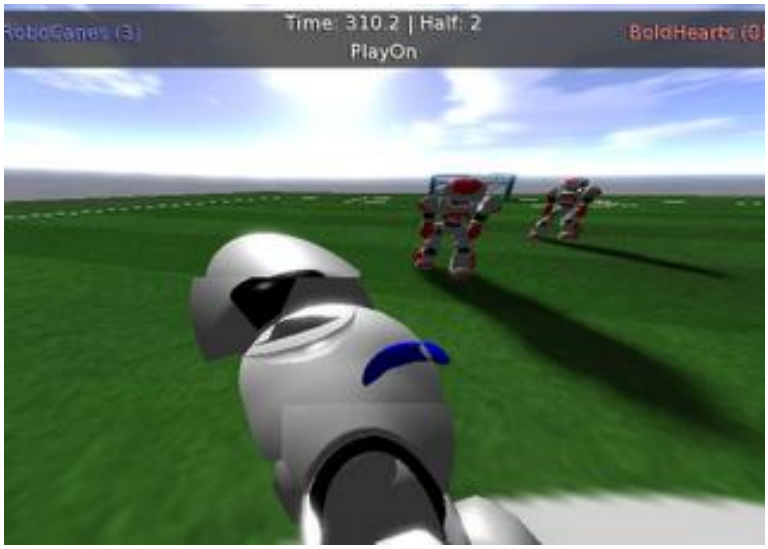
Niektoré objekty je možné počas simulácie manuálne presúvať na iné pozície. Je to veľmi užitočné na testovanie správania, lokalizácie a rozhodovania.

Všetky objekty, ktoré je možné v RoboViz manuálne presúvať majú okolo seba farebnú kružnicu. Sú to hráči a lopta. Tieto predmety môžu byť presunuté na ľubovoľné miesto na hracej ploche.

Obrázok č.6: Presúvateľný objekt

2.2.7. Pohľad z kamery robota

Základný pohľad je klasicky z hora z vtáčej perspektívy, kde sa je možné pohybovať. RoboViz ponúka ešte jednu možnosť a to je pohľad z kamery hráča.



Obrázok č.7: Pohľad z kamery robota

2.2.8. Vykresľovací protokol

Jednou z primárnych funkcií tohto programu je umožniť vykresľovať jednoduché útvary aj iným používateľom. Kvôli tomu obsahuje RoboViz jednoduchý sieťový protokol pre klientov na kontrolu vykresľovania. Klienti spolupracujú s Roboviz pomocou príkazov, na vykreslenie určitých útvarov. Príkazy sa posielajú cez sieť pomocou protokolu UDP. Používateľ posiela príkazy do RoboViz, ktoré

sa následne vykresľujú, tiež je možné zobrazovať iba určité vlastnosti pre jednotlivých hráčov. Pri požiadavke na vykreslenie útvarov sa príkaz rozloží na jednotlivé podpríkazy. RoboViz používa dva zásobníky, predný a zadný, pričom tieto príkazy vloží do zadného zásobníka. Tieto príkazy sa ešte nevykresľujú, ale čaká sa na príkaz na načítanie príkazu do predného zásobníka. Pri obdŕžaní tohto príkazu sa obsah zadného zásobníku premiestni do predného. Predný zásobník vždy obsahuje príkazy, ktoré sa majú vykresliť. Zadný zásobník sa potom vyprázdni. Útvary, ktoré RoboViz vykresľuje sa delia na statické a animované. Statické sú také, ktoré už po načítaní nemusia byť aktualizované, ako napríklad mriežka na hracej ploche. Animované sú také, ktoré sa musia aktualizovať, ako napríklad smer pohybu robota. Roboviz používa pre koordináciu systém zo simulačného servera SimSpark.

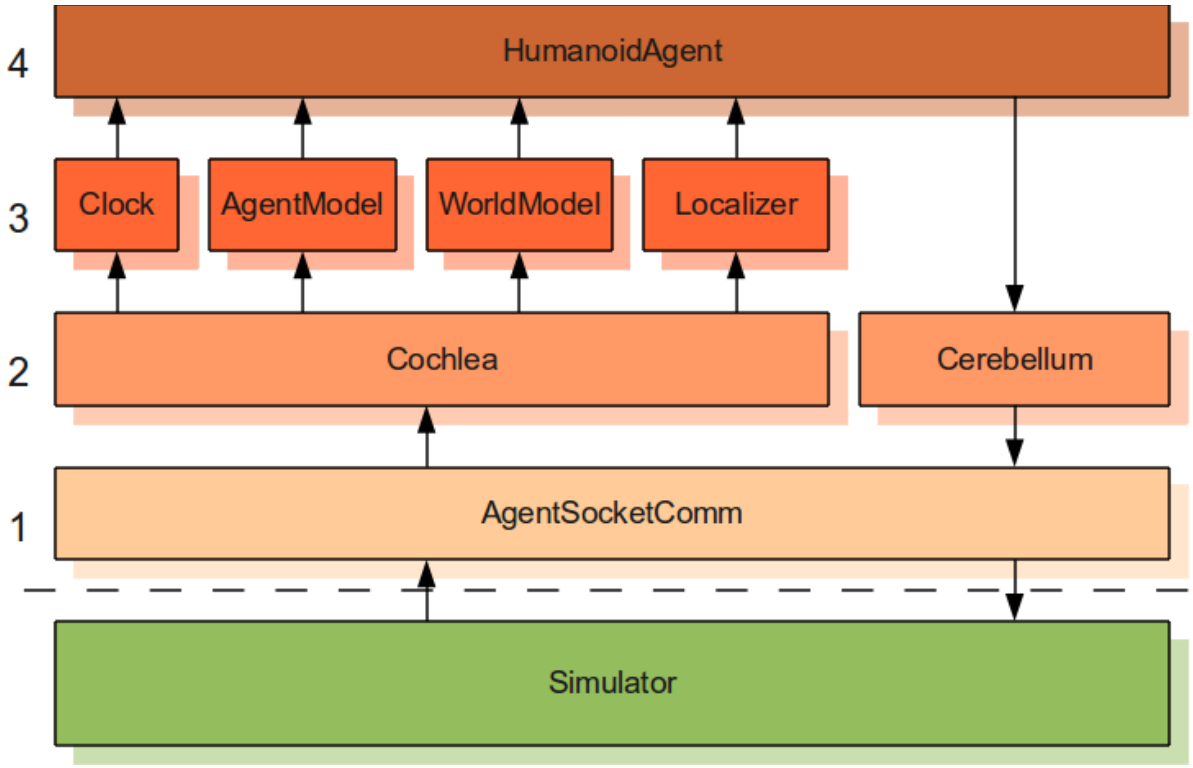
Zdroj: https://sites.google.com/site/umroboviz/

2.3. Little Green BATS

Little Green BATS je holandský tím z univerzity v Groningene. Agent tohto tímu sa skladá z niekoľkých častí a vrstiev. Všetky nižšie vrstvy sú nezávislé od vyšších.

Spodná úroveň komunikácie medzi simulátorom a agentom používa ASCII cez S-výrazy a TCP/IP spojenie. Modul *AgentSocketComm* zabezpečuje nastavenia tohto spojenia, vytváranie a spracovanie správ a ich transformáciu z a do ľahšie spracovateľnej dátovej štruktúry.

Integrácia vstupu a výstupu je zabezpečená pomocou druhej vrstvy na trochu vyššej úrovni. Na strane vstupu modul *Cochlea* získa všetky dáta zo stále textovej správy vytvorenej modulom *AgentSocketComm* a transformuje ich do použiteľnejšie binárne hodnoty. Modul *Cerebellus* sa používa na zhromažďovanie riadiacich príkazov, pričom z nich v prípade nevyhnutnosti odstráni protichodné príkazy a posunie ich v textovej štruktúre modulu *AgentSocketComm*, ktorý im v tomto tvare rozumie.



Obrázok č.8: Moduly libbats

Modely. Štyri moduly v tretej vrstve používajú dáta z modulu *Cochlea* na aktualizovanie modelov do súčasného stavu sveta ako sú aktuálny čas, stav v akom sa nachádza telo agenta, stav sveta a hry a rozostavenie všetkých objektov na ihrisku.

Inteligencia je implementovaná v najvyššej vrstve. Inštancia *HumanoidAgent* má prístup ku všetkým informáciám nazhromaždených v rozličných moduloch, na základe ktorých posiela akcie do modulu *Cerebellus.*

2.1.1. Jednotlivé triedy

AgentSocketComm

Najnižšia trieda knižnice, ktorá zabezpečuje komunikáciu so serverom pomocou TCP soketov. Tiež zabezpečuje spojenie so serverom, posielanie, prijímanie a spracovanie správ. Trieda je navrhnutá ako singleton. Udržuje dve vnútorné fronty pre správy. Jednu pre vstupné a jednu pre výstupné správy. Sú postupne napĺňané a vyprázdňované pri volaní metódy *update() triedy AgentSocketComm*. Vykoná sa v prípade, že boli prijaté nové dáta zo serveru. Ako bolo spomínané zabezpečuje tiež odosielanie správ na server a spracovanie prijatých správ.

Cochlea

Je to vrstva nad *AgentSocketComm*, ktorá vyťahuje informácie z dát získaných zo servera a upravuje ich na jednoducho prístupnú formu. Používa vlastné mená pre kĺbové senzory, ktoré sa nezhodujú s názvami používanými serverom. Vnútorné mená sú „head1“, „head2“, „larm1“ až „larm4“ pre ľavú ruku a „rarm1“ až „rarm4“ pre pravú ruku, „lleg1“ až „lleg6“ a „rleg1“ až „rleg6“ pre nohy. Server

používa mená ako „laj1“ ,čo je prvý kĺb na ľavej ruke, alebo „rlj3“ čo je zas tretí kĺb na pravej nohe. Modul *Cochlea* využíva preklad týchto názvov na vlastné, čo následne umožňuje ovládať aj rozdielne modely robotov.

Clock

Vracia aktuálny čas.

AgentModel

Udržuje informácie o vlastnom stave agenta. Pozíciu kĺbov, dáta zo senzorov a tiež dáta vyššej zložitosti ako napríklad ťažisko. Pri inicializácii načíta XML konfiguračný súbor s menom, veľkosťou a váhou častí tela agenta a kĺbov. Tieto dáta využíva tiež na nastavenie názvov pre modul *Cochlea*, takže to nie je potrebné robiť ručne. Po inicializácii je agent pripravený na použitie.

WorldModel

Dáta ponúkané z *Cochlea* nemusia byť priamo použiteľné, pretože môžeme potrebovať informácie, ktoré zo získaných dát vyplývajú. Práve toto robí *WorldModel*. Ponúka napríklad aktuálny stav hry alebo veľkosť ihriska, ale tiež aj informácie na vyššej úrovni ako kto útočí, alebo či je iný hráč bližšie k lopte. Na začiatku *WorldModel* taktiež potrebuje vedieť názov tímu pre niektoré funkcie. Následne sa model aktualizuje v každom cykle. *WorldModel* získava dáta z *Cochlea* ak boli aktualizované pred aktualizovaním *WorldModel*.

Localizer

Localizer je abstraktná trieda od ktorej sa odvodzujú rôzne implementácie. Jednou z nich je *KalmanLocalizer*, ktorý udržuje pozíciu hráča, lopty, ostatných hráčov, bránky a rohových zástaviek. Systém používa viacero koordinačných pozícií.

- **Pozícia agenta** znazorňuje jeho pozíciu, pričom stred je v jeho ťažisku. Kladná os x smeruje paralelne s pravým ramenom, kladná os y priamo pred telo agenta a kladná os z smerom do stredu hlavy agenta. Tento systém používa *AgentModel* na koordinovanie častí tela.

Obrázok č.9: Pozícia agenta

- **Lokálna pozícia**. Stred tiež súradnicovej sústavy je tiež v ťažisku hráča, akurát že kladná os z smeruje vždy priamo hore a osi x a y ležia rovnobežne s čiarami ihriska. Túto sústavu používa *Localizer* na orientáciu a určenie pojmov ako „predo mnou“, „vedľa mňa“, „nado mnou“.

Obrázok č.10: Lokálna pozícia

- **Globálna pozícia** je plne nezávislá od pozície agenta. Jej stred leží v strede ihriska. Kladná os x smeruje do stredu súperovej bránky, kladná os y vľavo pri pohľade v smere osi x a kladná os z nahor kolmo na osi x a y. Toto je druhá sústava, ktorú používa *Localizer* a je najlepšia na určenie globálnych pomerov.

Obrázok č.11: Globálna pozícia

Cerebellum

Spája výsledky z viacerých zdrojov a vytvára akcie, pričom odstraňuje protichodné príkazy. Aktuálne má k dispozícií 5 rôznych akcií a to:

- *MoveJointAction*: pohni jednoduchým kĺbom alebo zložitým v jednej osi
- *MoveHingeJointAction*: pohni jednoduchým kĺbom
- *MoveUniversalJointAction*: pohli zložitým kĺbom v smere všetkých osí
- *BeamAction*: nasmeruj sa na určitú pozíciu
- *SayAction*: povedz niečo

Po vyskladaní akcie, sa zavolá *outputCommands()* na odoslanie akcie cez *AgentSocketComm*.

HumanoidAgent

Zabezpečuje všetko potrebné pre fungovanie agenta.

Zdroj: http://launchpadlibrarian.net/60052249/libbatsmanual-2.0.1.pdf

2.4. B-human

Nemecký tím B-human pochádza z univerzity v Brémach a Robocup-u v štandardnej platforme sa účastní od roku 2008. Aktuálne sú trojnásobný víťazi Robocup-u v štandardnej platforme pre rok 2009,2010 a 2011. Všetky informácie vychádzajú z dokumentácie tímu. [1]
Ich výskum je prevažne zameraný na inverznú kinematiku a udržanie stability hráča pri vykonávaní jednotlivých pohybov pomocou sledovania ťažiska hráča (COM – center of mass).

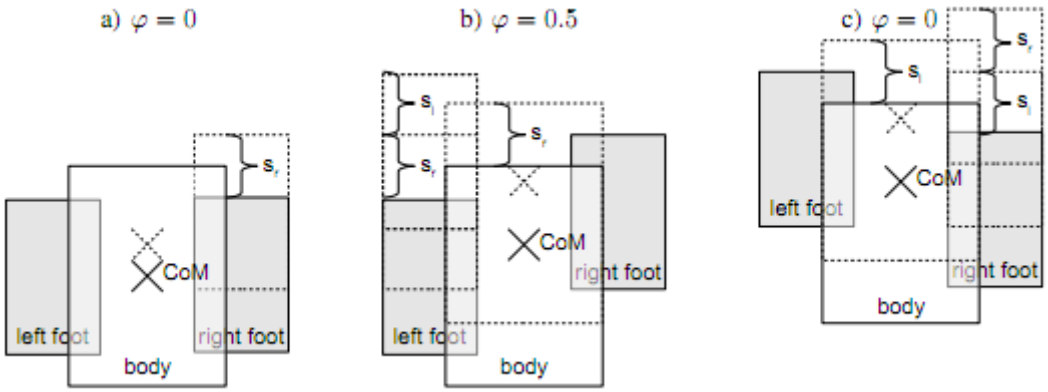
2.4.1. *Inverzná kinematika*

Inverzná kinematika sa zaoberá výpočtom pohybov kĺbov hráča potrebných na dosiahnutie koncového stavu. Analytické riešenie tejto problematiky nie je priamočiare. Pohyb nôh je opísaný maticami homogénnych transformácií, ktoré obsahujú rotácie a posuny nôh. V dokumentácii tento tím uvádza pomerne rozsiahly postup, ako vytvorili rovnice vedúce k výpočtu týchto matíc, berúc do úvahy šesť kĺbov na každej nohe hráča.

2.4.2. *Chôdza a kop*

Pri modelovaní chôdze a kopov sa využíva inverzná kinematika, no predovšetkým sa sleduje pohyb ťažiska, aby sa docielila maximálna stabilita hráča aj na úkor jeho rýchlosti. Pri chôdzi platí, že ťažisko v polohe robota po vykonaní jedného kroku by nemalo opustiť fyzický model robota v stave pred vykonaním tohto kroku. Tomuto sa musí prispôsobiť predovšetkým dĺžka krokov. Pri kope zasa platí, že ťažisko hráča sa na začiatku kopu má presunúť do vertikálnej línie nohy, na ktorej hráč počas kopu stojí. V dokumentácii tohto tímu sa nachádza množstvo rovníc opisujúcich pohyb ťažiska pri vykonávaní rôznych pohybov.

[1] http://www.b-human.de/publications/



Obrázok č.12: Poloha ťažiska v jednotlivých fázach chôdze.

2.4.3. Roly hráčov

Ďalšou zaujímavosťou, ktorou sa tento tím zaoberá je dynamické prideľovanie rolí hráčom na ihrisku. Majú zadefinované štyri roly, ktoré určujú správanie sa hráča na ihrisku:

Útočník – jeho úlohou pohybovať sa predovšetkým na polovici súpera. Prioritou je ísť k lopte a vystreliť na bránku alebo prihrať, pokiaľ je uhol príliš malý a tým pádom nízka šanca na skórovanie. Pokiaľ je bránka príliš ďaleko, alebo je v ceste prekážka, tak si útočník loptu posúva.

Záložník – jedinou úlohou záložníka je podpora útočníka, čo znamená, že sa záložník drží v určitej vzdialenosti za útočníkom a pokiaľ je bližšie k lopte ako útočník, tak sa k nej snaží dostať a nahrať ju útočníkovi.

Obranca – obranca správa podobne ako brankár, ale nesmie vstúpiť do pokutového územia. Pohybuje sa v jednej línii po šírke ihriska a snaží sa postaviť do jednej línie s bránou, loptou a súperovým hráčom najbližšie k lopte tak, aby sa v tejto línii nenachádzal brankár, teda brankár mal "voľný výhľad".

Brankár – je jediný hráč, ktorý sa pohybuje v pokutovom území a pokiaľ sa z nejakého dôvodu nachádza mimo tohto územia (napr. dynamická zmena rolí) je jeho primárnou úlohou vrátiť sa do bránkoviska. Inak sa snaží postaviť medzi súperovho hráča s loptou a bránku, pričom vykonáva špecifické obranné pohyby.

Tieto roly sa dynamicky menia podľa aktuálneho stavu prostredia. Pokiaľ sa ktorýkoľvek hráč nachádza v pozícii kopu do lopty, tak sa z neho stáva automaticky útočník.

Pokiaľ sa hráč nachádza najbližšie k bráne, stáva sa z neho brankár atď. Tieto zmeny rolí prispievajú k lepšej spolupráci hráčov a taktickému pohybu hráčov po ihrisku.

http://www.b-human.de/publications/

2.5. UT Austin Villa

Tento americký tím Texaskej univerzity v Austine pôsobí na scéne Robocup-u od roku 2003 a patrí medzi najúspešnejšie tímy v štandardnej lige, 3D simulovanej lige a lige trénerov. Na svojich domovských stránkach majú zverejnené rozsiahle kódy k svojim agentom spolu s dokumentáciou.[2] Ich výskum sa zameriava na trénovanie, hracie stratégie a v poslednom roku aj na optimalizáciu všesmerovej chôdze (angl. omnidirectional walk), ktorú označujú za hlavný dôvod ich víťazstva na majstrovstvách sveta v simulovanom futbale v Turecku. Detaily parametrov použitých na optimalizáciu tejto chôdze však momentálne nie sú zverejnené.

2.5.1. Trénovanie

Princíp trénovania uplatňuje tento tím vo vývoji dvoch druhov agentov:

Tréner (angl. couch) – je to agent schopný na základe záznamov odohratých zápasov analyzovať hru súpera. Po analyzovaní gólových situáciách vie navrhnúť taktiku, ktorá by tomuto gólu zabránila a naopak po analyzovaní obrany vie navrhnúť taktiku, pri ktorej by bolo možné obranu obísť a skórovať.

Hráč (angl. couchable player) – agent schopný spracovať pokyny trénera a následne ich vykonať. Hráč zohľadňuje aj ručne pridané rady konfiguračného charakteru (napr. uprednostniť streľbu pred prihrávaním, posúvať loptu viac dopredu než dozadu, pri obrane posúvať loptu k najbližšej čiare ...)

Tréner analyzuje dáta pozostávajúce z cyklicky zaznamenávaného stavu prostredia. Tento stav obsahuje pozície, smer a rýchlosť pohybu lopty a všetkých hráčov. Výsledkom analýzy je identifikácia udalostí vyššej úrovne, ktoré sa počas hry udiali (držanie lopty, strela, prihrávka, získanie lopty ...). Úlohou trénera je následne na základe daného stavu prostredia určiť najpravdepodobnejšiu udalosť a postup, ktorý majú hráči v prípade tejto udalosti vykonať. Tento postup je potom vykonávaný hráčom, na základe rozhodovacieho stromu, ktorý je implementovaný v tréningovom jazyku CLang.

[2] http://www.cs.utexas.edu/~AustinVilla/?p=home

2.5.2. Hracie stratégie

Hracie stratégie určujú správanie agentov pri určitých herných situáciách. Tím UT Austin Villa vyvinul dve podúlohy robotického futbalu, ktoré sú zamerané na kontrolu lopty.

Keepaway – v tejto úlohe figurujú dva tímy, jeden sa snaží kontrolovať loptu a udržať ju na ihrisku obmedzenej veľkosti a druhý tím sa snaží prebrať kontrolu lopty, pričom veľkosť ihriska a počet hráčov oboch tímov je voliteľný. Táto úloha je náročná z pohľadu strojového učenia, pretože:

- stavový priestor je príliš veľký na úplné preskúmanie
- každý agent má len čiastočné informácie o aktuálnom stave
- priestor akcií je kontinuálny
- viacero spoluhráčov sa musí učiť súčasne

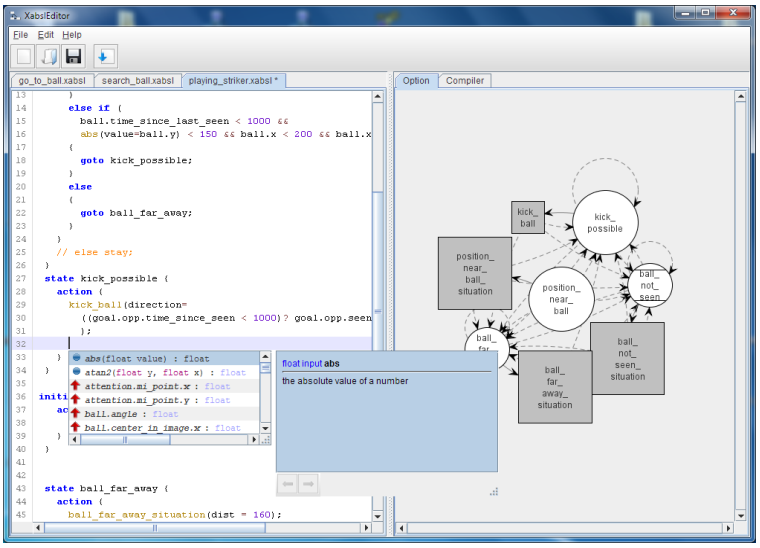
Half field offense – je rozšírením *keepaway*, kedy sa útok jedného tímu má za úlohu dostať cez obranu druhého tímu tak, aby bol schopný streliť gól. Hrá sa na jednej polovici ihriska so začiatočnou pozíciou lopty blízko poliacej čiary.

2.6.Nao Team Humboldt

Tento nemecký tím bol založený v roku 2007 v Berlíne. S robotom Nao začali pracovať v máji roku 2008 a hneď v tomto roku dosiahli 4. miesto na turnaji v Suzhou. Taktiež dosiahli 3. miesto v technickej súťaži v Grazi v roku 2009. Tím NTH nezverejňuje kompletnú dokumentáciu ale každý rok zverejňuje Team Descrition Paper a rôzne ďalšie publikácie. Tento tím má už základné pohyby zvládnuté na veľmi vysokej úrovni, a preto sa momentálne sústredí na plynule prepojenie dvoch rôznych pohybov ako napr. chôdze a kopu. Členovia tohto tímu skúmali aplikáciu Constraint based techniques a porovnávali probabilistickú metódu Monte Carlo s Kalmanovými filtrami. V dokumentáciách sa spomínajú dve zaujímavé softvérové aplikácie a to XabslEditor a Simple Soccer Agent.

XabslEditor je grafický editor pre "Extensible Agent Behavior Specification Language" XABSL. Ide o open source softvér implementovaný v jazyku Java a spustiteľný na všetkých bežných platformách. Jazyk XABSL je jednoduchý jazyk pre opis správania autonómnych agentov pomocou konečných stavových automatov. Tento jazyk bol vyvinutý práve pre návrh správania robotov pre Robocup. Keďže stránka hovorí, že tímy používajúce

tento jazyk dosiahli viaceré úspechy v Robocupe bolo by vhodné tento jazyk bližšie zanalyzovať.



Obrázok č.13: Xabsl Editor

Druhou aplikáciou je Simple Soccer Agent. Táto aplikácie je príkladom použitia architektúry tímu NTH a beží na oficiálnom RoboCup 3D simulátore. Program je vhodný pre začiatočníkov a je implementovaný veľmi minimalisticky a preto poskytuje dobrý základ pre vlastné experimenty.

2.7. Diplomová práca – robotický brankár

Študent Peter Ertl sa vo svojej diplomovej práci zaoberá tvorbou prototypu brankára a zaoberá sa všetkými jeho aspektami ako je inverzná kinematika, nižšie správanie hráča, vyššie správanie hráča a učenie hráča. Analýza v tomto projekte je veľmi podrobná a poskytuje takmer všetky dôležité informácie o problémovej oblasti.

Študent navrhol zjednotenie správania a pohybov pod jedným rozhraním. Navrhnutý model umožňuje vkladanie nových riešení ako modulov, a to konkrétne pohyby, správanie a nové implementácie prechodových funkcií v stavovom automate. V ďalšej časti návrhu sa zaoberá modelom lopty. Táto časť je veľmi podrobná a zaoberá sa správaním lopty v simulačnom prostredí. Ďalej študent zjednodušil formát pohybov, vynechaním niektorých irelevantných častí a implementoval niekoľko pohybov brankára. Taktiež navrhol modul pre skladanie pohybov, aby mohli viaceré pohyby pristupovať ku kĺbom. Každý pohyb má priradenú kladnú váhu a ak dva pohyby posúvajú ten istý kĺb, jeho výsledné posunutie je

určené váženým priemerom. Študent sa tiež zaoberal v návrhu predikciou správania súpera avšak podarilo sa mu implementovať len jednoduchú predikciu, kedy brankár sleduje, či útočník zastal a mieri na bránku v snahe dať gól.

Táto diplomová práca je na veľmi slušnej úrovni, obsahujúca zaujímavé informácie a je dobrým základom pre pokračovanie v práci na brankárovi simulačného robotického futbalu.

2.8. Kouretes Robocup Team

Tím Kouretes vznikol vo Februári 2006. Spočiatku sa tím zameriaval výhradne na „štvornohú ligu“ (the four-legged league), ale neskôr svoje aktivity rozšíril na simulačnú ligu (the simulation league). V súčasnosti je tím aktívny v Standard Platform League.,

V roku 2008 zahájil tím vývoj kódu pre novú platformu (Nao robot) v kombinácii C++ a Ruby pre reálneho robota, C# pre simulovaného robota v MSRS (Microsoft Robotics Studio) a Java pre simulovaného robota Webots.

- 3. miesto v Nao lige
- 1. miesto v MSRS simulácii
- Najlepšia osmička vo Webots RobotStadium súťaži

V roku 2009 sa tím zameral výhradne na Nao robotov a vyvinul nové moduly používajúce C# a Python.

- Účasť na SPL RoboCup German Open 2009
- Účasť na súťaži RoboCup 2009 v Rakúsku
- 6. miesto vo Webots simulácii

V roku 2010 vyvinuli nové moduly pre kontrolu pohybov, komunikácie, rozpoznávaniu objektov, seba-lokalizáciu, vyhýbanie sa prekážkam a tímový kód bol zverejnený na úložisku Github.

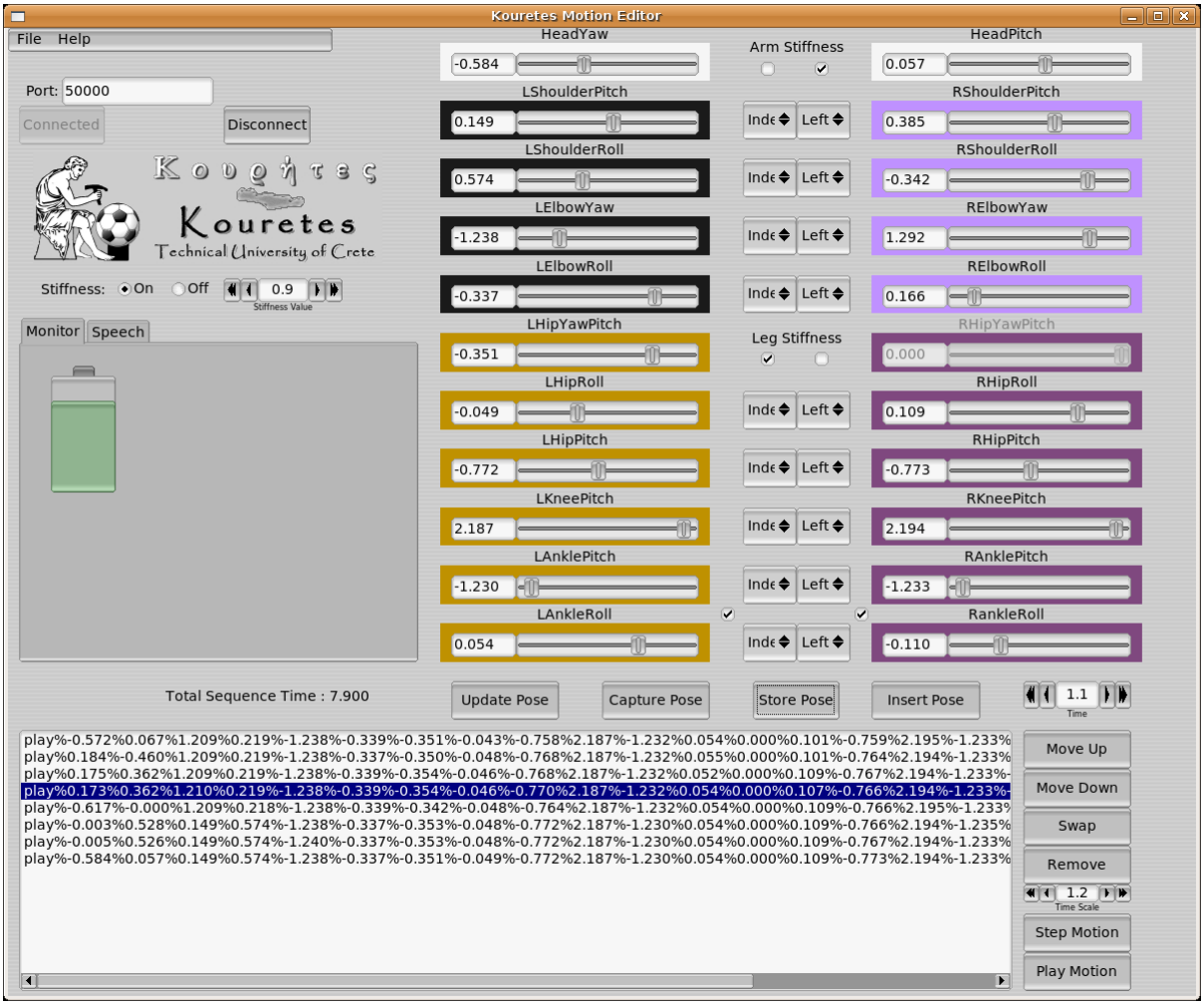
- Účasť na prvej súťaži RoboCup Mediterranean Open v Ríme
- Účasť na RoboCup 2010 v Singapure.
- Hostili prvý oficiálny SPL turnaj v Grécku (pozvané 3 tímy)

V roku 2011 boli vyvinuté ďalšie nové moduly.

- Spojenie s anglickým tímom Noxious pre RoboCup 2011
    - 1 výhra a 3 prehry
- 2. miesto na SPL Open Challenge súťaži

2.8.1. *Komplexné pohybové vzorky*

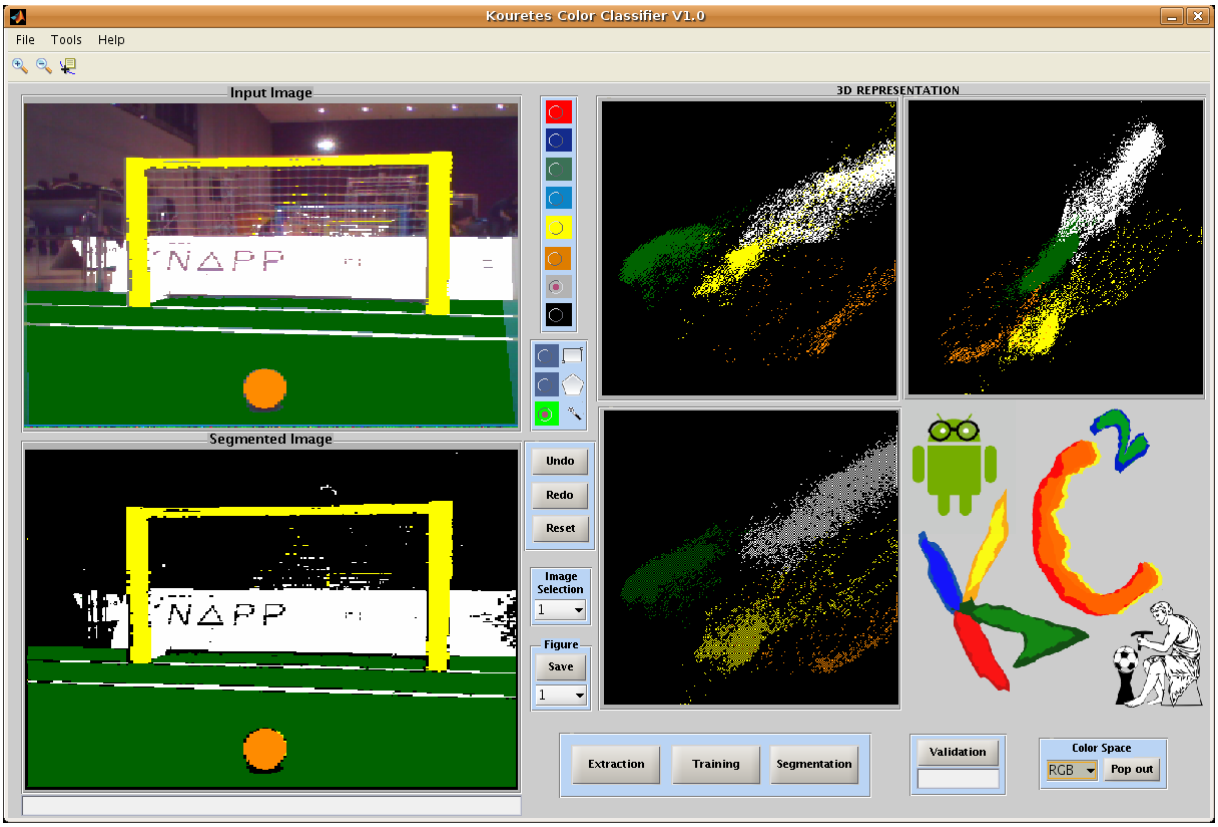
Kouretes Motion Editor (KME) je interaktívny nástroj pre vytváranie komplexných pohybov.



Obrázok č.14: Kouretes Motion Editor

2.8.2. *Vision*

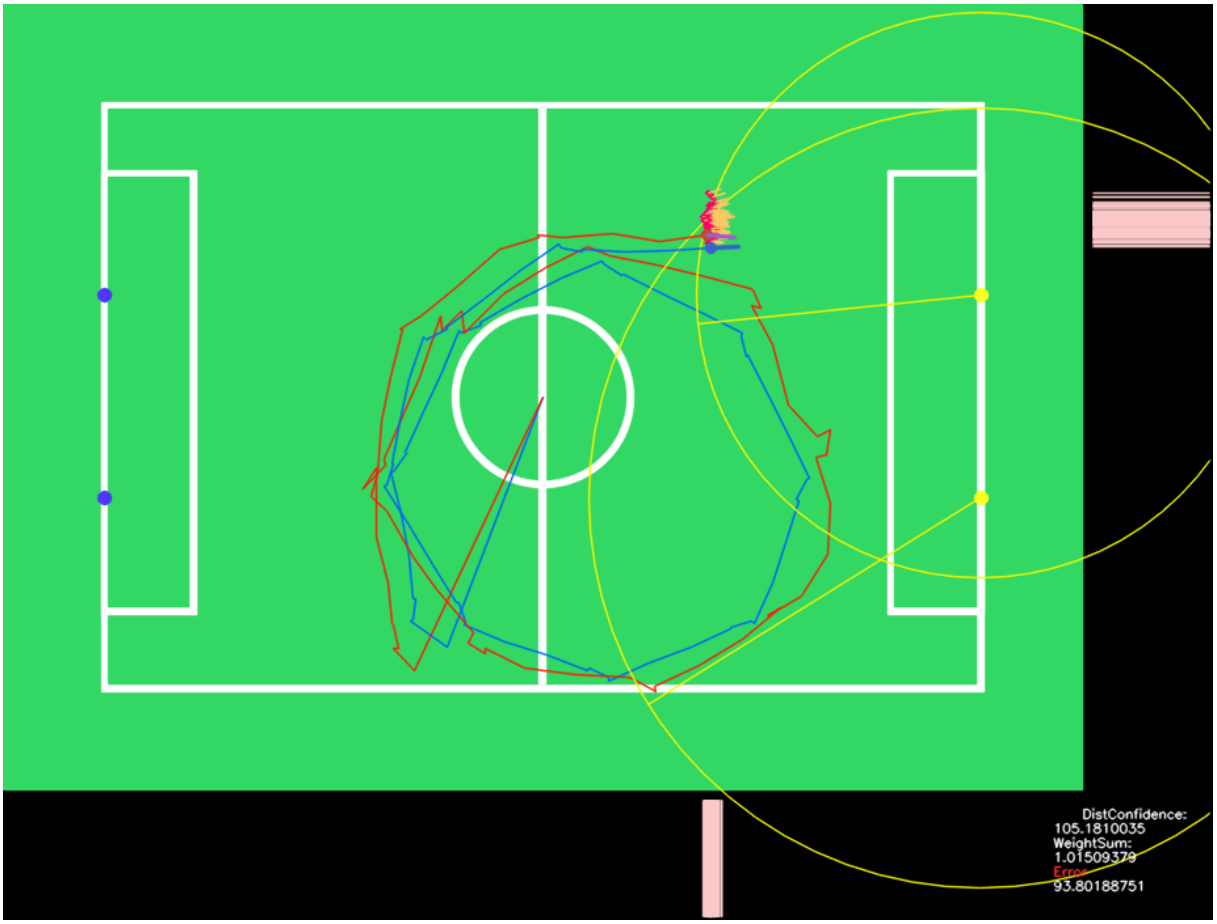
Kouretes Color Classifier (KC^2) je grafický nástroj pre označovanie objektov a učenie sa farebnej segmentácii.



Obrázok č.15: Kouretes Color Classifier

2.8.3. Kouretes Localization

Kouretes Localization (KLoc) je plne funkčný a parametrický modul pre robotovu sebalokalizáciu. Vyvinuli jednoduchú metódu pre rýchle učenie v rôznych situáciách.



Obrázok č.16: Couretes Localization

2.8.4. Skill Learning

Využili podporné učenie pre naučenie sa pohybu kopu počas státia na jednej nohe bez pádu. Po 150 opakovaniach sa postupne naučil tento pohyb, ktorý bol plynulý a synchronizovaný.

Obrázok č.17: Skill learning

2.8.5. Záver

Tím Kouretes sa neumiestňuje na popredných priečkach výsledkových listín svetových súťaží, no namiesto toho vsadil na kvalitnú prípravu, ktorá spočíva na vytváraní vlastných modulov/nástrojov na vytváranie pohybov robota, jeho učenie, sebalokalizáciu a rozpoznávanie objektov.

2.9. Tím Zigorat

Tím pozostáva z ľudí z viacerých iránskych univerzít. Zverejňujú svoj základný kód, čím chcú pomôcť iným ľuďom urýchliť ich vstup do tejto oblasti – ako píšu na svojich stránkach, príliš veľa času sa totiž často strávi veľmi low-level vecami.

Ich kód je nezávislý na modeli robota (tzn. tento model je dobre oddelenou súčasťou, ktorá sa dá ľahko nahradiť), je objektovo orientovaný a veľmi dobre zdokumentovaný.

Tím postupne pracoval s 3 modelmi hráčov - „legged sphere“, tj. guľa na nohách, neskôr s robotom SoccerBot a napokon s oficiálnym modelom robota Nao.

Ich kód sa skladá z 3 vrstiev:

- komunikačnej – stará sa o komunikáciu medzi agentom a SimSparkom
- dátovej – má na starosti spracovávanie prijatých dát a na základe nich vytváranie približného modelu sveta
- rozhodovacej – robí rozhodnutia a vracia ich späť komunikačnej vrstve

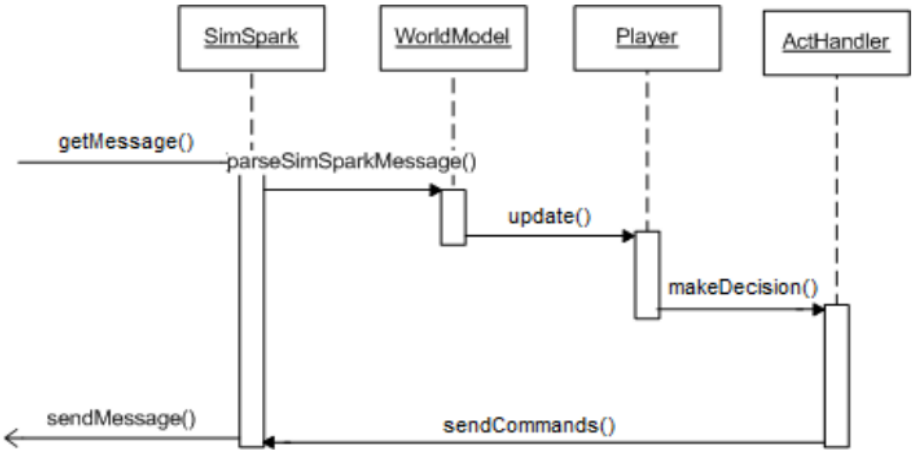


Fig. 3: Sequence diagram

Obrázok č.18: Sekvenčný diagram fungovania aplikácie

Spravovanie modelu sveta uloženého v každej inštancii agenta má na starosti trieda WorldModel, ktorá je navrhnutá tak, že aj v prípade pridania nových perceptorov dokáže fungovať bez zmien – umožňuje teda jednoduché prechody na nové verzie.

Schopnosti (skilly) sú implementované pomocou podtried triedy Skill, kde každá schopnosť je samostatná trieda, ktorá má prístup ku kompletnému stavu agenta.

Súčasťou agentov je taktiež podpora pre logovanie.

ZigoBot Designer

Dôležitým nástrojom, ktorý tento tím vytvoril a využíva je ZigoBot designer. Jedná sa o editor scén podporujúci rôznych robotov (rovnako ako agent). Scéna je napísaná v S-výrazoch a zložená z uzlov. Príklady uzlov tvoria napríklad rôzne perceptory, efektory, transformácie atď.

Používateľské rozhranie umožňuje okrem editácie aj priamy náhľad na robota z ľubovolného uhla. Zdrojový kód programu je podobne ako kód agenta podrobne zdokumentovaný.

Zdroj:
http://sites.google.com/site/zigorat3d

2.10. Analýza tímu Nao-Team HTWK Leipzig

2.10.1. O tíme

Tím vznikol vo februári 2009 na Univerzite aplikovaných vied v Leipzigu. Tím sa zúčastnil turnaja RoboCup German Open 2009 a obsadil druhé miesto. Obsadil prvé miesto v súťaži Technical Challenge of RoboCup 2009. V roku 2010 sa dostal do štvrťfinále RoboCup v Singapúre a získal prvé miesto v súťaži Open challenge počas tohto turnaja.

Tím skúma mnohé oblasti úzko späté s robotickým futbalom. Ide hlavne o spracovanie obrazu a jeho segmentáciu, lokalizáciu samotného robota, pohyby a ich vytváranie, architektúru softvéru a optimalizáciu vnorených systémov s obmedzenými zdrojmi. Niektoré vybrané riešenia sú uvedené v podkapitolách.

2.10.2. Segmentácia

Tím v začiatkoch používal na identifikáciu ihriska a objektov len na základe tabuliek farieb, podľa ktorých jednotlivé objekty určoval. Táto metóda však mala veľkú nevýhodu v zmene jasu, pričom je generovanie takejto tabuľky pri zmenených jasových podmienkach veľmi časovo náročné. Ďalším problémom bolo, že na jednotlivých súťažiach sa menili aj farby lopty.

Kvôli týmto problémom sa rozhodol tím používať „real-time“ segmentáciu bez potreby kalibrovania. Vyvinuli algoritmus na segmentáciu obrazu, pričom dokážu rozoznávať tvary objektov, pričom nie je potrebné tento algoritmus kalibrovať pri zmene jasových podmienok. Základom algoritmu sú malé historamy založené na tabuľkách farieb, ktoré sa automaticky generujú počas segmentácie každého obrázku. Tieto tabuľky sa vytvárajú iteratívne po identifikovaní objektu pomocou jeho tvaru alebo odhadnutej farby.

Tabuľka č.1: Časy jednotlivých fáz segmentácií

Fáza	Čas[ms]
Rozpoznanie podlahy	2
Rozpoznanie čiary	8
Rozpoznanie lopty	5
Rozpoznanie gólu	9

Táto metóda bola úspešne použitá na turnajoch German Open 2010 a RoboCup 2010, kde vyhrala cenu Open Challenge.

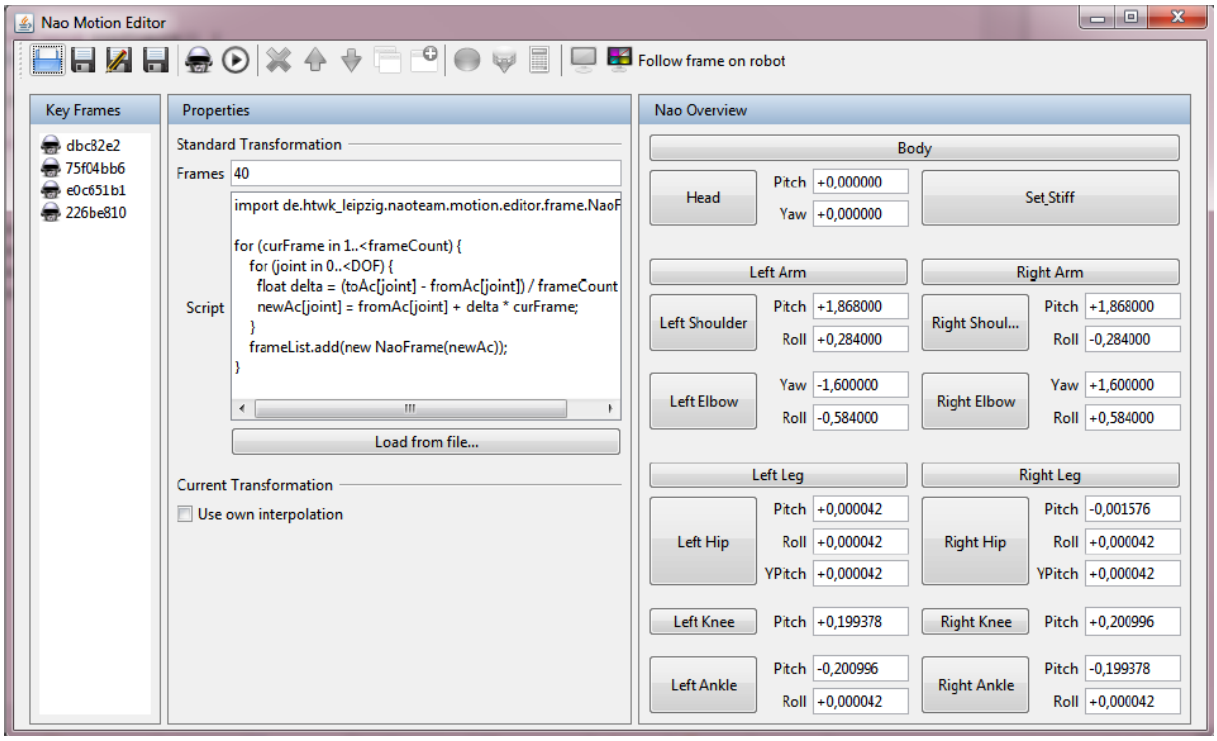
2.10.3. Chôdza

Do začiatku roka 2010 používal tento tím genetický algoritmus na pohyb chodenia. Tieto pohyby boli síce rýchle, ale nie aplikovateľné na všetky smery. Kvôli tomuto nedostatku sa rozhodli vyvinúť nový spôsob chôdze, ktorý je založený na parametrizovateľnom modeli chôdze a podporovaný novým stabilizačným algoritmom. Výhodou tohto algoritmu je možnosť veľmi rýchlej chôdze do všetkých smerov s dodržaním potrebnej stability. Tento algoritmus plánujú zverejniť po jeho úspešnej integrácii.

Algoritmus je podobný, ako v [1], pričom sa krok parametrizuje smerom chôdze, rýchlosťou chôdze a uhlovou rýchlosťou. Tento prístup dosahuje nízku výpotovä zložitosť.

2.10.4. Editor pohybov

Základným cieľom je získanie kľúčových rámcov od robota , narábanie s nimi a interpolácia s Groovy scripting engine medzi nimi. Existuje predefinovaný skript, ktorý definuje lineárnu a vyhladzovaciu interpoláciu medzi rámcami. Tieto zachytené pohyby sú uložené v XML súbore a môžu byť exportované do formátu iného tímu. Architektúra editora je navrhnutá tak, aby bolo rýchle pridať novú funkcionalitu, pričom sa teda nové požiadavky môžu vytvárať rýchlo podľa potreby.



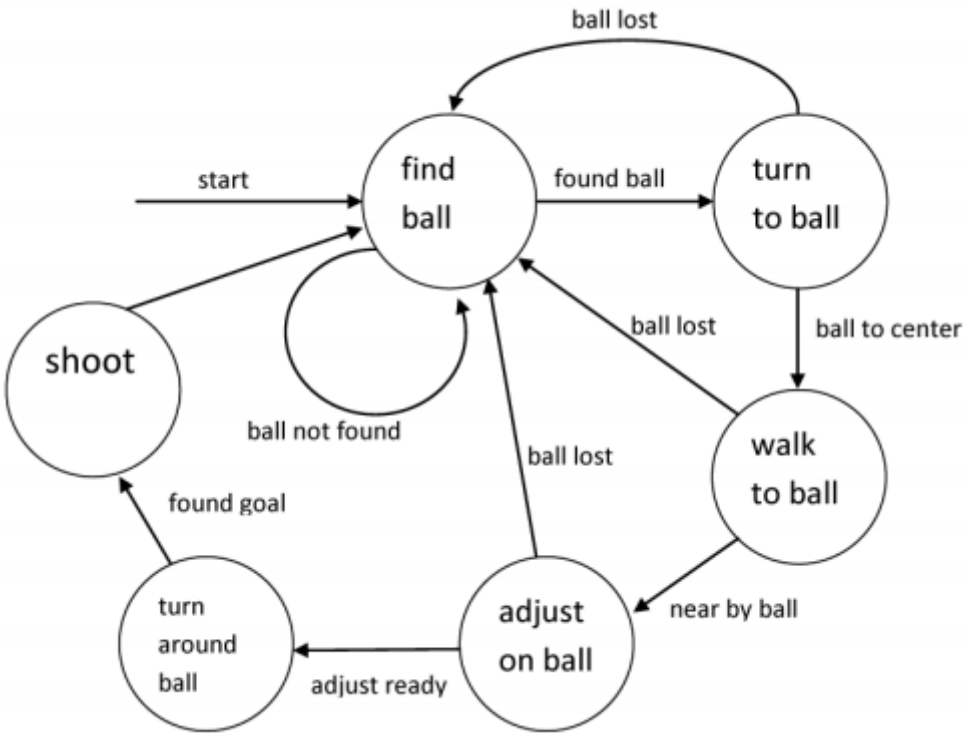
Obrázok č.19: Editor pohybov

2.10.5. Stratégia

Stratégia hry je založená na konečnom stavovom automate, ktorý prechádza šiestimi stavmi:

- nájdi loptu
- otoč sa k lopte
- kráčaj k lopte
- nastav sa na loptu
- otáčaj sa okolo lopty
- strieľaj

Každý z týchto stavov zahŕňa sériu pohybových a segmentačných nastavení, napríklad v prípade hľadania lopty sa segmentácia zameriava len na loptu, pričom ignoruje ciele a ostatné objekty. V prípade otáčania sa okolo lopty je zas segmentácia zameraná na ciele, pričom sa neprihliada na aktuálnu polohu lopty, keďže je predpoklad, že je v blízkosti nohy. Takéto zmýšľanie podporuje zrýchlenie rozhodovania a aj rýchlosť vykonávania oboch stavov, pričom sa dosahuje vyššia reakčná rýchlosť a minimalizuje sa otáčanie hlavou.



Obrázok č.20: Konečný stavový automat hráča

Odkazy na literatúru

[1] Sven Behnke. Online trajectory generation for omnidirectional biped walking. Proceedings 2006. IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., pages 1597 – 1603, May 2006.

[2] http://naoteam.imn.htwk-leipzig.de

2.11. *Analýza fyzikálneho modelu*

2.11.1. Server

Server SimSpark je fyzikálny simulačný multiagentový systém pre agentov v troj-dimenzionálnom priestore. Sever je vytvorený v jazykoch C++ a Ruby a na vizualizáciu sú použité dve knižnice OpenGL a SDL.

2.11.2. Perceptory

Na vnímanie sveta v RoboCup 3D slúžia perceptory. Server zasiela agentovi informácie o samotnom agentovi, videných objektoch a podobne.

GyroRate slúži na opis orientácie tela robota vzhľadom k jednotlivým osiam. Tento perceptor je umiestnený v hornej časti robota Nao.

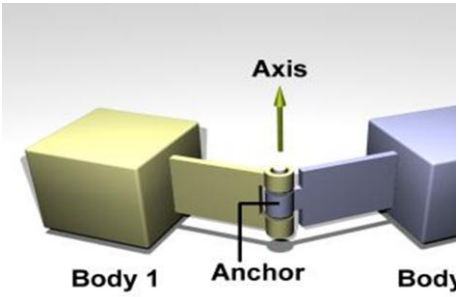
Tabuľka č.2: Opis orientácie tela robota vzhľadom k osiam

Formát správy:	(GYR (n <name>) (rt <x> <y> <z>)
<name>	Časť tela
<x> <y> <z>	Uhlová rýchlosť v smere troch osí voľnosti v stupňoch za sekundu
Príklad:	(GYR (n torso) (rt 0.01 0.07 0.46))

HingeJoint určuje o koľko stupňov je ktorý kĺb natočený.

Tabuľka č.3: Opis natočenia kĺbu

Formát správy:	(HJ (n <name>) (ax <ax>))
<name>	Identifikátor kĺbu
<ax>	Uhol natočenia kĺbu
Príklad:	(HJ (n laj3) (ax J1.02)



Obrázok č.21: Anatómia prepojenia kĺbu

ForceResistance perceptor slúži na oznámenie o pôsobiacej sile na časť tela a jej vektore. Správa obsahuje dvoje súradnice, jedny určujú bod, na ktorý sila pôsobí a druhé súradnice určujú vektor tejto sily

Tabuľka č.3: Opis správy na oznámenie o pôsobiacej sile

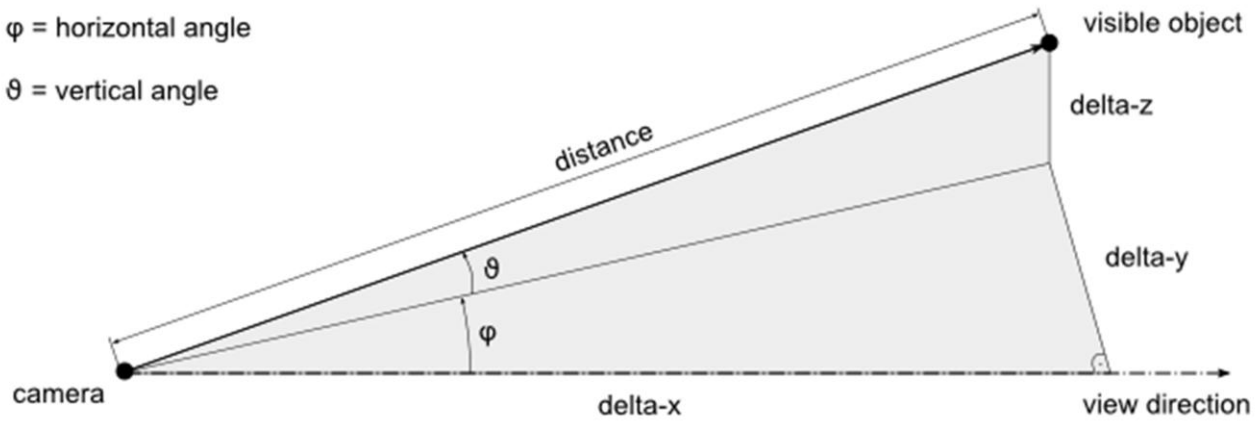
Formát správy:	(FRP (n <name>) (c <px> <py> <pz>) (f <fx> <fy> <fz>))
<name>	Časť tela
<px> <py> <pz>	Súradnice, kde nastalo pôsobenie sily
<fx> <fy> <fz>	Súčasti vektora sil
Príklad:	(FRP (n lf) (c J0.14 0.08 J0.05) (f 1.12 J0.26 13.07))

Accelerometer perceptor umiestnený v hornej časti tela počíta zrýchlenie.

Tabuľka č.4: Opis zrýchlenia

Formát správy:	(ACC (n <name>) (a <x> <y> <z>))
<name>	Časť tela
<x> <y> <z>	Súčasné zrýchlenie
Príklad:	(ACC (n torso) (a 0.00 0.00 9.81)

Vision perceptor dostáva informácie o objektoch videných agentom (ostatní agenti, lopta, statické body ihriska). Zorné pole robota Nao je 120°.



Obrázok č.22: Rozpoznávanie objektov vision perceptrónom

Chyby vnášané do vision perceptora:

- Malá kalibračná chyba je pridávaná do pozície kamery. Pre každú os z intervalu -0.005 m až 0.005 m. Chyba je vypočítaná raz a zostáva nemenná počas celého zápasu.
- Dynamický šum s normálnym rozdelením okolo 0
- Chyba vzdialenosti: $\sigma = 0.0965$
- Chyba φ: $\mu = 0.1225$
- Chyba ϑ : $\mu = 0.1480$

GameState vysiela niekoľko základných informácií o aktuálnom stave hry.

Tabuľka č.5: Opis stavu hry

Formát správy:	(GS (t <time>) (pm <playmode>))
<time>	Čas od rozohrávky
<playmode>	(GS (t 0.00) (pm BeforeKickOff))
Príklad:	(ACC (n torso) (a 0.00 0.00 9.81)

HearPerceptor – priama komunikácia medzi robotmi nie je povolená, ale môžu si vymieňať správy cez simulačný server. Na túto komunikáciu slúži tento perceptor.

Tabuľka č.6: Opis výmeny správ

Formát správy:	(hear <time> 'self'|<direction> <message>)
<time>	Čas počutia správy
<direction>	Smer odkiaľ správa prišla
<message>	Samotná správa
Príklad:	(hear 12.3 self helloworld) (hear 12.3 -12.7 helloyourself)

Na tento perceptor sa vzťahujú nasledujúce obmedzenia:

- Maximálna dĺžka správy je 20 bytov
- Správa môže obsahovať iba znaky <0x20, 0x7E> okrem medzery a obyčajných zátvoriek
- Správa vyslaná zo vzdialenosti viac ako 50m nie je „počutá"
- Každý hráč môže v jednom okamihu počuť najviac jednu správu z každého tímu. Perióda obnovovania senzoru je 0,4 sekundy. Ak príde viacero správ toho istého tímu, sú spracované v poradí, v akom prišli a zvyšné správy sú ignorované. Hráč počuje svoje správy vždy.

2.11.3. Efektory

Create – v momente ako je agent pripojený na server nie je viditeľný a nemôže zasiahnuť do simulácie. Pomocou Create hráč odovzdá serveru cestu ku konfiguračnému súboru, ktorý definuje fyzickú reprezentáciu agenta a množinu jeho perceptorov a efektorov.

Tabuľka č.7: Opis odovzdania konfiguračného súboru

Formát správy:	(scene <filename>)
<filename>	Relatívna cesta ku konfiguračnému súboru
Príklad:	(scene rsg/agent/nao/nao.rsg)

HingeJoint efektor dávajúci príkaz na ohnutie kĺbu.

Tabuľka č.8: Opis ohybu kĺbu

Formát správy:	(<name> <ax>)
<name>	Identifikátor kĺbu
<ax>	Uhol natočenia
Príklad:	(lae3 5.3)

Beam efektor určuje umiestnenie agenta na hracej ploche pred začatím polčasu

Tabuľka č.9: Opis umiestnenia agenta

Formát správy:	(beam <x> <y> <rot>)
<x>, <y>	Počiatočné súradnice
<rot>	Natočenia agenta
Príklad:	(beam 10.0 -10.0 0.0)

Say efektor slúži na odoslanie správy pre komunikáciu medzi hráčmi.

Tabuľka č.10: Opis komunikácie medzi hráčmi

Formát správy:	(say <message>)
<message>	Posielaná správa
Príklad:	(say helloworld)

Init priradí hráča k tímu a pridelí mu číslo. Ak agent posiela ako číslo 0, server priradí hráčovi najbižšie voľné číslo automaticky

Tabuľka č.11: Opis iniciálizácie hráča

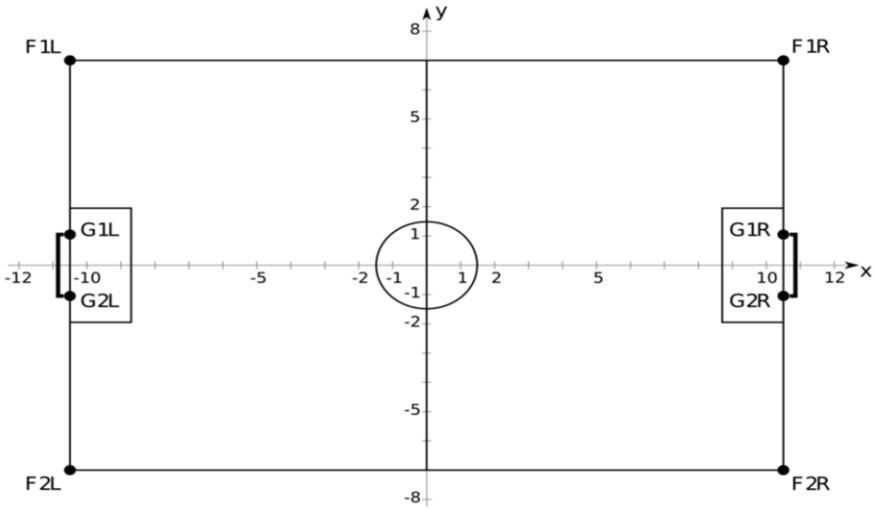
Formát správy:	(init (unum <playernumber>)(teamname <yourteamname>))
<playernumber>	Číslo hráča
<yourteamname>	Meno tímu
Príklad:	(init (unum 1)(teamname FHO))

http://simspark.sourceforge.net/wiki/index.php/Models#Perceptor_and_Effector_Names

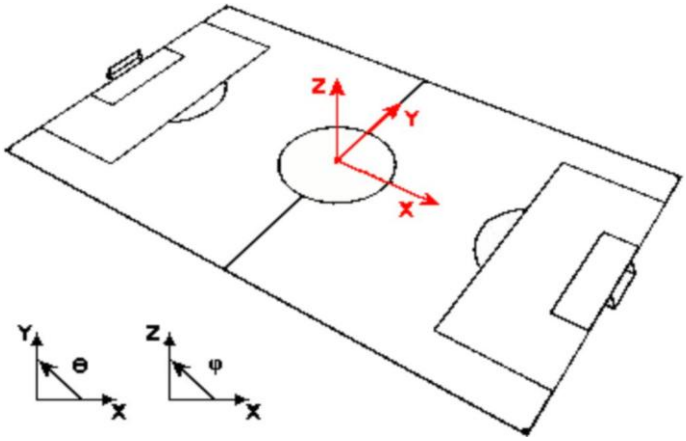
2.11.4. Ihrisko

Šírka ihriska je 14 metrov a jeho dĺžka 21 metrov (pozn. Platí pre najnovšiu verziu serveru, predtým boli rozmery 18x12 metrov). V strede je kruh s priemerom 1.5 metra. V skutočnosti ale nejde o skutočný kruh, ale je vymodelovaný z niekoľkých krátkych úsečiek. Bránky sú široké 2.1 metra, vysoké 0.8 metra a hlboké 0.6 metra. Rozmery bránkoviska sú 3.9m x 1.8m. Ihrisko je ohraničené hranicou širokou 10 metrov v každom smere. Po tejto ploche mimo ihriska sa agent pohybovať nemôže.

Ihrisko obsahuje 8 orientačných statických bodov rozpoznateľných agentmi. Štyri sa nachádzajú v rohoch ihriska (F1L, F2L, F1R, F2R) v nulovej výške (súradnica Z – 0.0). Ďalšie 4 sa nachádzajú na bránkach (G1L, G2L, G1R, G2R), konkrétne na žŕdkach vo výške 0.8 metra.



Obrázok č.23: Schéma futbalového ihriska s rozmiestnením statických bodov



Obrázok č.24: Orientácia osí vzhľadom na ihrisko

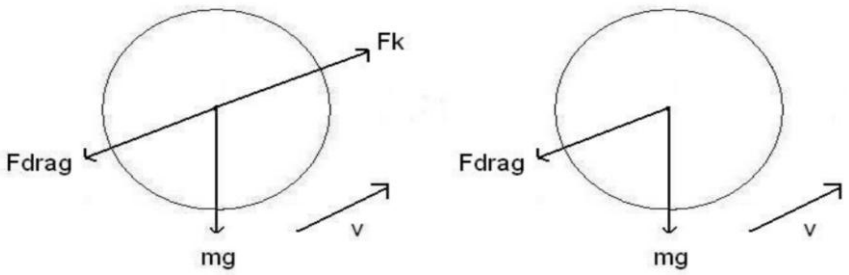
http://simspark.sourceforge.net/wiki/index.php/Main_Page

2.11.5. Lopta

Lopta nachádzajúca sa v simulačnom prostredí má rádius 0.04 metra a jej váha je 26 gramov.

Fyzika pri pohybe lopty

Pohyb lopty je rozdelený do dvoch fáz.



Obrázok č.25: Prvá a druhá fáza pohybu lopty

Prvá fáza:

Vektor sily pri pohybe lopty je definovaný ako $\vec{F_k} = \langle F_{kx}, F_{ky}, F_{kz} \rangle$, kde

$$\sum F_x = \cos\varphi_k \cos\theta_k - \mu_B v_x = m_B a_x$$
$$\sum F_y = \cos\varphi_k \sin\theta_k - \mu_B v_y = m_B a_y$$
$$\sum F_z = \sin\varphi_k - m_{Bg} - \mu_B v_z = m_B a_z$$

Druhá fáza:

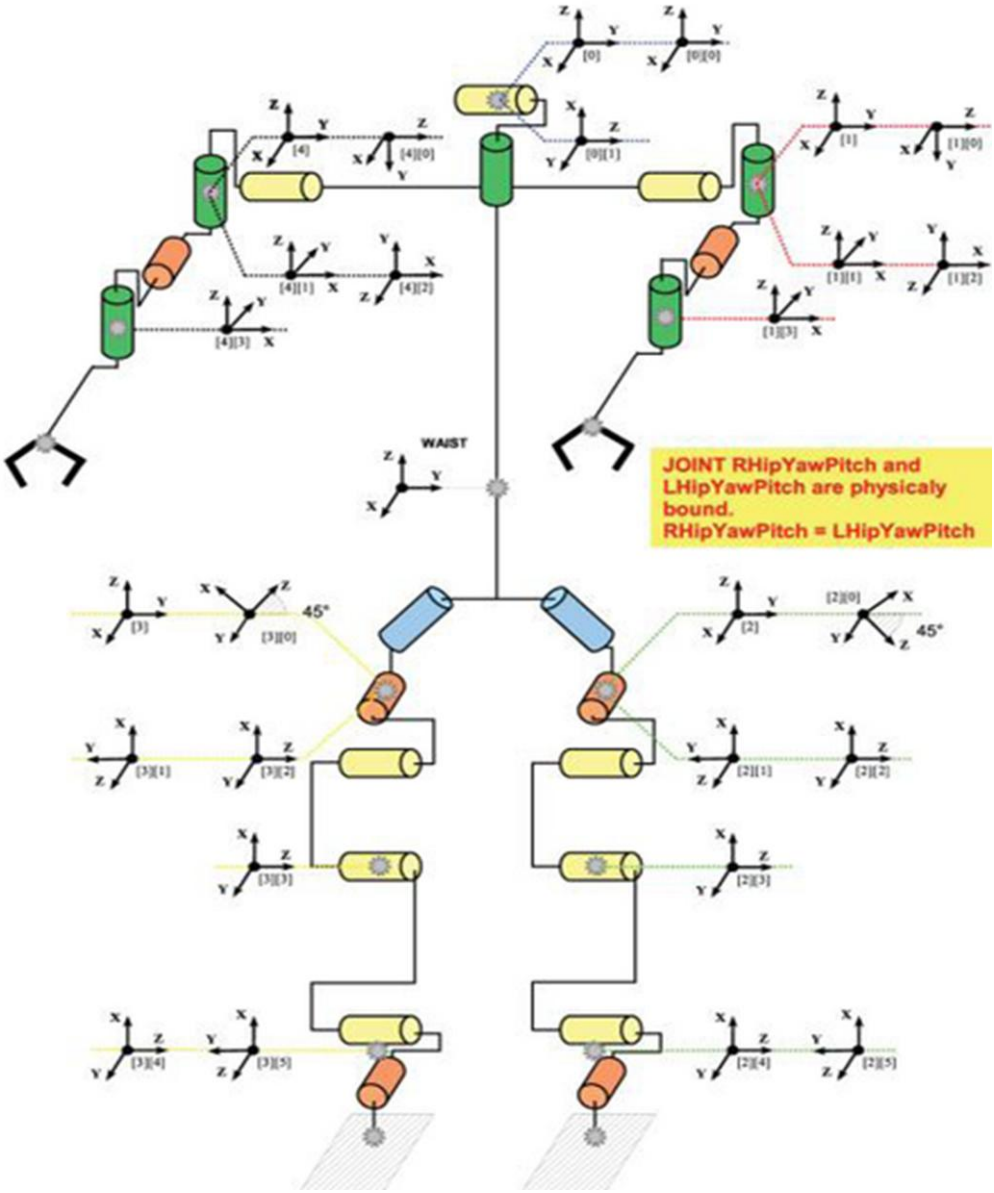
Celkové sily sú $\sum F_x = -\square_B v_x = m_B a_x$
$\sum F_y = -\square_B v_y = m_B a_y$
$\sum F_z = m_{Bg} - \square_B v_z = m_B a_z$

Súčiniteľ odporu pre loptu $\square_B \approx 0.214785 \frac{kg}{s}$

http://www.eng.auburn.edu/SCS-TM/P18.pdf

2.11.6. Agent

Hráčov (agentov) reprezentujú humanoidné roboty Nao. Ich výška je približne 57 cm a váha asi 4.5 kg. Nao má 22 kĺbov, 6 v každej nohe, 4 v každej ruke a 2 v krku.

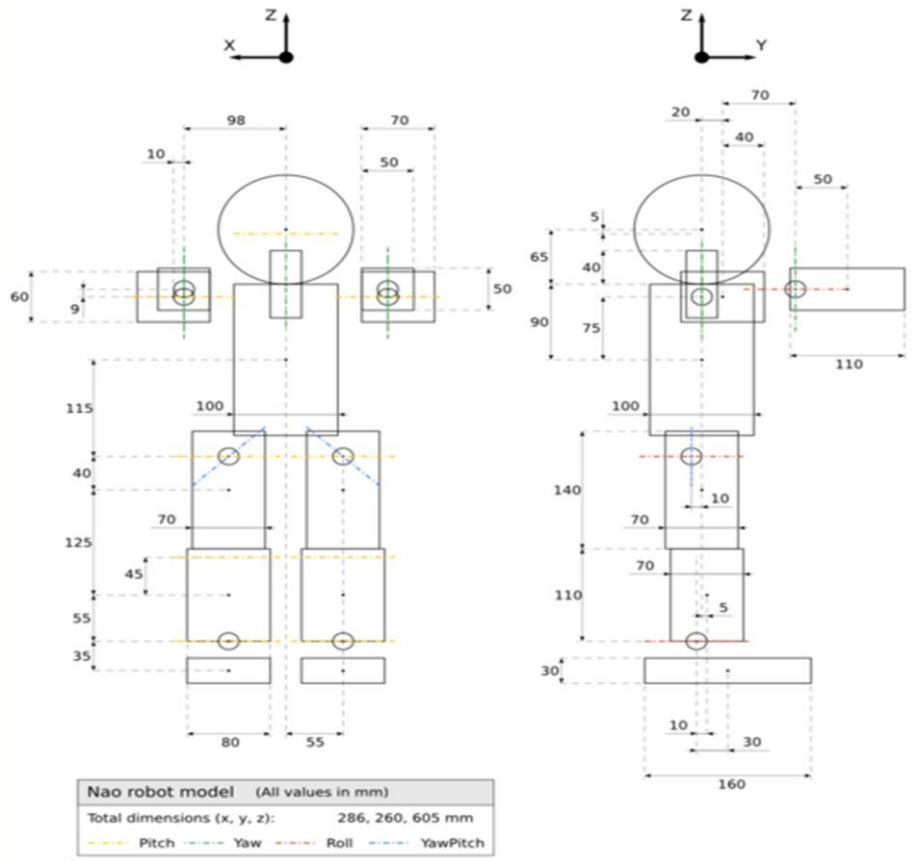


Obrázok č.26: Anatómia robota NAO

Tabuľka č.12: Zoznam kĺbov a ich identifikátorov pre robota NAO

No.	Description	Hinge Joint	Perceptor name	Effector name
1	Neck Yaw	[0][0]	hj1	he1
2	Neck Pitch	[0][1]	hj2	he2

3	Left Shoulder Pitch	[1][0]	laj1	lae1
4	Left Shoulder Yaw	[1][1]	laj2	lae2
5	Left Arm Roll	[1][2]	laj3	lae3
6	Left Arm Yaw	[1][3]	laj4	lae4
7	Left Hip YawPitch	[2][0]	llj1	lle1
8	Left Hip Roll	[2][1]	llj2	lle2
9	Left Hip Pitch	[2][2]	llj3	lle3
10	Left Knee Pitch	[2][3]	llj4	lle4
11	Left Foot Pitch	[2][4]	llj5	lle5
12	Left Foot Roll	[2][5]	llj6	lle6
13	Right Hip YawPitch	[3][0]	rlj1	rle1
14	Right Hip Roll	[3][1]	rlj2	rle2
15	Right Hip Pitch	[3][2]	rlj3	rle3
16	Right Knee Pitch	[3][3]	rlj4	rle4
17	Right Foot Pitch	[3][4]	rlj5	rle5
18	Right Foot Roll	[3][5]	rlj6	rle6
19	Right Shoulder Pitch	[4][0]	raj1	rae1
20	Right Shoulder Yaw	[4][1]	raj2	rae2
21	Right Arm Roll	[4][2]	raj3	rae3
22	Right Arm Yaw	[4][3]	raj4	rae4



Obrázok č.27: Box model robota NAO

Fyzika pri pohybe hráča

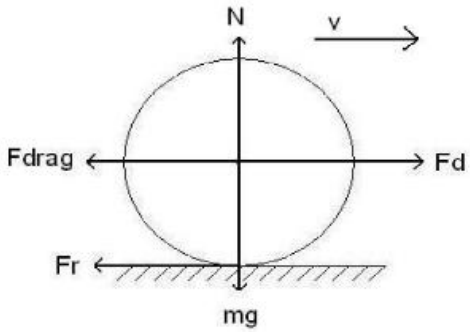
Vektor sily pri pohybe hráča je definovaný ako $\vec{F}_d = \langle F_{dx}, F_{dy} \rangle$,

kde $F_{dx} = F_d \cos \alpha \quad F_{dy} = F_d \sin \alpha$

Celkové sily sú $\sum F_x = F_d \cos \alpha - k_A v_x = m_A a_x$
$\sum F_y = F_d \sin \alpha - k_A v_y = m_A a_y$

Súčiniteľ odporu pre agenta $k_A \approx 190.65 \frac{kg}{s}$

Odporová sila je definovaná ako $F_{drag} = -kv \quad kde \quad k = 6\pi r \eta, \quad \eta = 0.1026 \frac{kg}{m \cdot s}$



Obrázok č.28: Fyzika pri pohybe hráča

http://www.eng.auburn.edu/SCS-TM/P18.pdf

Tabuľka č.13: Detailný popis konfigurácie robota NAO

Name	Parent	Translation	Mass	Geometry	Name	Anchor	Axis	Min	Max
torso			1.2171	Box 0.1, 0.1, 0.18					
neck	torso	0, 0, 0.09	0.05	Cylinder L: 0.08 R: 0.015	HJ1	0, 0, 0	0,0,1	-120	120
head	neck	0, 0, 0.065	0.35	Sphere 0.065	HJ2	0, 0,-0.005	1,0,0	-45	45
shoulder	torso	0.098, 0, 0.075(r) -0.098, 0, 0.075(l)	0.07	Sphere 0.01	AJ1	0, 0, 0	1,0,0	-120	120
upperarm	shoulder	0.01, 0.02, 0(r) -0.01, 0.02, 0(l)	0.150	Box 0.07, 0.08, 0.06	AJ2	-Translation	0,0,1	-95(r) -1(l)	1(r) 95(l)
elbow	upperarm	-0.01, 0.07, 0.009(r) 0.01, 0.07, 0.009(l)	0.035	Sphere 0.01	AJ3	0, 0, 0	0,1,0	-120	120
lowerarm	elbow	0, 0.05, 0	0.2	Box 0.05, 0.11, 0.05	AJ4	-Translation	0,0,1	-1(r) -90(l)	90(r) 1(l)
hip1	torso	0.055, -0.01, - 0.115(r) -0.055, -0.01,- 0.115(l)	0.09	Sphere 0.01	LJ1	0, 0, 0	- 0.7071,0,0.70 71(r) -0.7071,0,- 0.7071(l)	-90	1
hip2	hip1	0, 0, 0	0.125	Sphere 0.01	LJ2	0, 0, 0	0,1,0	-45(r) -25(l)	25(r) 45(l)

thigh	hip2	0, 0.01, -0.04	0.275	Box 0.07, 0.07, 0.14	LJ3	-Translation	1,0,0	-25	100
shank	thigh	0,0.005,-0.125	0.225	Box 0.08, 0.07, 0.11	LJ4	0,-0.01, 0.045	1,0,0	-130	1
ankle	shank	0, -0.01,-0.055	0.125	Sphere 0.01	LJ5	0, 0, 0	1,0,0	-45	75
foot	ankle	0, 0.03,-0.035	0.2	Box 0.08, 0.16, 0.03	LJ6	-Translation	0,1,0	-25(r) -45(l)	45(r) 25(l)

2.12. Vytvorenie Wiki

K vytvoreniu wiki tímu viedlo rozhodnutie používať wiki formát na zdieľanie a zverejňovanie dokumentov medzi členmi tímov. Rozhodli sme sa ho použiť najmä vďaka možnosti organizácie dokumentov do určitej štruktúry a ich jednoduchého vzájomného odkazovania. Samozrejmou vlastnosťou je uchovávanie kompletnej histórie zmien dokumentov a možnosť prezerať si zmeny medzi jednotlivými verziami. Wiki používali oba tímy.

Použitý bol softvér MediaWiki, hlavne kvôli tomu, že je široko rozšírený a väčšina ľudí sa s ním už stretla. Wiki beží na serveri tímu 17, http://vm08.ucebne.fiit.stuba.sk/wiki

Všetci členovia z oboch tímov majú práva správcov, vytvorené stránky sú zamknuté pre editovanie neregistrovanými a novo registrovanými používateľmi z dôvodu zamedzenia vandalizácie spambotmi.

Okrem nasadenia wiki bola taktiež vytvorená hlavná stránka so stručným návodom a základnou hierarchickou štruktúrou, do ktorej sa majú dokumenty vkladať. Počiatočné rozdelenie bolo na analýzy a technické dokumenty (špecifikácie, návrhy a pod.).