Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A

INFORMAČNÝCH TECHNOLÓGIÍ

Tímový projekt

RoboCup 2D – Nové stratégie

Inžinierske dielo

Tím č. 4 Kukuričné deti

Bc. Jozef Balga

Bc. Jaroslav Bálik

Bc. Peter Holotík

Ing. Bc. Rastislav Kršák

Bc. Tomáš Kučečka

Bc. Andrej Škuba

Študijný program: Softvérové inžinierstvo/Informačné systémy

Vedúci tímu: Ing. Marián Lekavý

Email: tim_04@googlegroups.com

Ročník: 1

Akademický rok: 2008/2009

Obsah

Zoznam tabuliek a obrázkov 11

1 Úvod 6

1.1 Opis častí dokumentu 6

1.2 Zadanie 6

2 Analýza súčasného stavu 8

2.1 Analýza tímov ktoré budeme vylepšovať 8

2.1.1 Tím UTTP 8

2.1.2 Tím Jahodoví princovia 10

2.1.3 Zhodnotenie výberu tímu, ktorý budeme vylepšovať 14

2.2 Analýza tímov pre výber vlastností 14

2.2.1 Brainstormers 14

2.2.2 Oxsy 17

2.2.3 Tsinghuaeolus 18

2.2.4 UvA Trilearn 2003 19

2.2.5 DSL United (2003) 23

2.2.6 Nexus2D 24

2.2.7 Dirty Dozen 25

2.2.8 Mainz Rolling Brains 26

2.2.9 YowAI 29
2.3 Opis soccer servera 30
2.3.1 Typy komunikácie medzi serverom a klientom 30
2.3.2 Informácie získané zo senzorov 30
2.4 Evolúcia soccer servra od verzie 9.0.4 až po 13.0.0 32
2.4.1 Verzia 9.0.4 32
2.4.2 Verzia 10.0.0 32
2.4.3 Verzia 11.0.0 32
2.4.4 Verzia 12.0.0 33
2.4.5 Verzia 12.1.0 33
2.4.6 Verzia 13.0.0 33
3 Špecifikácia 36
3.1 Špecifikácia požiadaviek 36
3.2 Špecifikácia vybranej funkcionality 38
4 Hrubý návrh 38
4.1 Správania, ktoré chceme prebrať 39
4.1.1 Streľba na bránu 39
4.1.2 Prihrávanie si hráčov 40
4.1.3 Driblovanie 40

4.1.4 Nadbiehanie hráča do voľného priestoru ... 41
4.2 Rozšírenia existujúceho hráča o zložitejšie vlastnosti ... 42
4.2.1 Modul pre posielanie správ medzi hráčmi ... 42
4.2.2 Posielanie správ medzi hráčmi ... 43
4.2.3 Voľba konkrétneho správania ... 44
5 Prototyp ... 45
5.1 Streľba na bránu ... 45
5.2 Drobné opravy kódu ako nástroj pre vylepšenie hráča ... 46
5.2.1 Vylepšenie výdrže obrancov ... 46
5.3 Komunikácia hráčov ... 48
5.4 Testovanie ... 49
5.4.1 Testovanie streľby na bránu ... 49
5.4.2 Testovanie vylepšenie výdrže obrancov ... 50
5.4.3 Testovanie komunikácie hráčov ... 50
5.5 Zhodnotenie prototypu ... 51
6 Opis riešenia ... 52
6.1 Výber implementačného prostredia a jazyka ... 52
6.2 Zmeny oproti návrhu ... 52
7 Realizácia riešenia ... 54

7.1 Zmena vo vykopávaní lopty brankárom 54

7.2 Hľadanie lopty hráčom 55

7.3 Rozšírenie modelu sveta 56

7.4 Správanie sa hráča s loptou 62

7.5 Prechod na server 13.0.2 63

8 Overenie riešenia 66

9 Zhodnotenie 68

9.1 Možné vylepšenia 69

Použitá literatúra 70

Zoznam príloh

Príloha A – Opis modelu sveta (World Model)

Príloha B – Opis správaní hráča

Príloha C – Návod na vytvorenie a použitie modulov správania

Príloha D – Inštalácia servera pod CygWin

Príloha E - Vytvorenie liveCD servera 13

Príloha F – Používateľská príručka

Príloha G – Vizualizácia rozhodovacích stromov správania sa hráča

Príloha H – Nahradené časti dokumentácie v letnom semestri

Zoznam tabuliek a obrázkov

Obrázok 1 Architektúra tímu UTTP 9

Obrázok 2 Architektúra hráča Jahodoví Princovia 12

Obrázok 3 Brainstormers- rozdelenie agenta na moduly 15

Obrázok 4 Brainstormers - Hierarchia správania[1] 16

Obrázok 5 Oxsy - "give-and-go"[2] 17

Obrázok 6 Oxsy - “Switch roles” [2] 18

Obrázok 7 Uvoľnenie: clear ball defensive, clear ball offensive, clear ball goal 22

Obrázok 8 Vrstvový model hráča Nexus2D [9] 24

Obrázok 9 Dvojfázové rozhodovanie sa hráča [9] 25

Obrázok 10 Vnútorný model hráča DirtyDozen [8] 26

Obrázok 11 Trojvrstvový model implementácie agenta [3] 27

Obrázok 12 Streľba na bránku [8] 39

Obrázok 13 Rozdelenie hracej plochy prostredníctvom priamok [8] 40

Obrázok 14 Rozhodovanie o smere a veľkosti odkopu lopty pri driblovaní 41

Obrázok 15 Nadbehnutie si hráča do voľného priestoru [2] 41

Obrázok 16 Návrh architektúry systému pre posielanie správ 42

Obrázok 17 Príklad komunikácie medzi spoluhráčmi tímu A 44

Obrázok 18 Streľba na bránku smerom k tyčke nezohľadňujúc pozíciu nepriateľa. 45

Obrázok 19 Streľba na bránku zohľadňujúc pozíciu nepriateľa. 46

Obrázok 20 Nelogické správanie sa obrancov. 47

Obrázok 21 Štruktúra posielania správ medzi hráčmi 48

1 Úvod

Tento dokument je jedným z výsledných produktov tvorivej činnosti tímu *Kukuričné deti* na predmete *Tvorba softvérového systému v tíme*. Jeho úlohou je zdokumentovať jednotlivé etapy vývoja softvéru – RoboCup hráča.

1.1 Opis častí dokumentu

Z formálneho hľadiska môžeme tento dokument rozdeliť na nasledujúce časti:

1. Úvod
2. Analýza existujúcich riešení (hráčov): domácich/fakultných aj zahraničných.
3. Špecifikácia požadovaných vlastností hráča a opis soccer servera.
4. Hrubý návrh riešenia.
5. Opis a testovanie prototypu.
6. Opis riešenia
7. Realizácia riešenia
8. Overenie riešenia
9. Zhodnotenie

1.2 Zadanie

Téme RoboCup, presnejšie lige simulovaného robotického futbalu, sa naši študenti venujú už deväť 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 možné dozvedieť sa na stránke STU turnaj v simulovanom robotickom futbale (www.fiit.stuba.sk/robocup).

V rámci fakulty sme realizovali viacero súťaží a posledné ročníky už boli oficiálnymi turnajmi iniciatívy RoboCup. Množstvo pozitívnych ohlasov nás priviedlo k vyhláseniu ďalšieho regionálneho turnaja RoboCup v simulovanej lige, opäť na záver akademického roka. Práve množstvo nových prístupov a riešení, ktoré predviedli nielen študenti tímového projektu, ale aj študenti umelej inteligencie, nám ukázalo, že možnosti na vylepšovanie hráča nie sú zďaleka vyčerpané a dokonca sa stále rodia prekvapujúce úspešné riešenia. Posledných niekoľko rokov sme preto ako podnázov použili „Nové stratégie". Znamená to všeobecne hľadanie nových prístupov a stratégií nielen pre hráča, ale aj vo svojej práci, v úpravách zdrojového kódu, podporných aplikáciách, základných aj vyšších schopnostiach hráča, spôsobe učenia a ladenia počas simulácií. Nové stratégie sú komplexnou výzvou do nového kola víťazstiev.

Na spresnenie je vhodné povedať, že v tomto tímovom projekte budeme rozširovať možnosti a vylepšovať správanie sa hráčov, vytvorených vo vlaňajších tímových projektoch. Využije sa existujúci zdrojový kód, dokumentácia a aj vytvorené podporné aplikácie. Musí sa tiež zachovať (a podľa možností aj zlepšiť) modularita a tým aj rozšíriteľnosť hráča. Zimný semester je vyhradený na oboznámenie sa s celým prostredím, najmä existujúcimi hráčmi a návrhu a prototypovej realizácii jeho vylepšení. Očakáva sa najmä návrh nových prístupov a stratégií vo všetkých už spomenutých oblastiach. Vybrané prístupy sa overia vytvorením jedného alebo viacerých prototypových rozšírení existujúceho kódu. Dôležitou súčasťou bude vytvorenie plánu implementácie a overovania nových stratégií v nasledovnom semestri. V letnom semestri nás čaká realizácia navrhnutých prístupov a stratégií a ich overovanie. Produkt by mal byť dohotovený v deviatom až desiatom týždni semestra, potom je potrebné venovať sa ladeniu a optimalizácii hráča na súťaž, ktorej výsledky sú súčasťou celkového hodnotenia tohto projektu.

2 Analýza súčasného stavu

V tejto kapitole robíme rozbor 2D prostredia simulačnej ligy RoboCupu a tiež už jestvujúcich tímov. Prvá časť sa venuje výberu hráča, ktorý poslúži ako základ pre vylepšovanie. V druhej časti vyberieme vlastnosti, ktorými budeme hráča vylepšovať. Na základe tejto analýzy vyberieme jednotlivé prvky hry hráčov, ktoré by mohli napomôcť pri zostavovaní nového hráča. Ďalej sú tu uvedené vlastnosti soccer servera a spôsoby komunikácie hráča so serverom.

2.1 Analýza tímov ktoré budeme vylepšovať

Táto kapitola sa venuje analýze tímov, ktoré by boli vhodné pre ďalšie rozšírenie v rámci tímového projektu RoboCup. Po vzájomnej konzultácii a po konzultácii s vedúcim nášho projektu sme sa rozhodli, že sa bude jednať o tímy vytvorené v rámci predošlých tímových projektov. Jedná sa konkrétne o tímy *UTTP* [5] a *Jahodoví Princovia* [6].

V nasledujúcich podkapitolách sú stručne zanalyzované tieto dva tímy. Bol vypracovaný opis ich architektúry a základné vlastnosti. V závere sú oba tímy zhodnotené spolu s výberom jedného z nich, ktorý považujeme za vhodnejší pre ďalšie rozšírenie.

2.1.1 Tím UTTP

Hráč tímu UTTP vychádza z hráčov tímu Loptoši, ktorý pôvodne vychádzali zo zdrojových súborov tímu FIITBA. Tím UTTP sa zúčastnil na RoboCup turnaji 2008 v rámci Slovenskej technickej univerzity. Podarilo sa mu dosiahnuť štvrté miesto z celkového počtu piatich miest.

Tímu UTTP sa podarilo rozšíriť pôvodného hráča hlavne o tieto vlastnosti:

- Modularizácia správania
- Spoločné úložisko údajov o prostredí hry
- Multiplatformovosť

Hráč komunikuje so serverom cez rozhranie, ktoré sprostredkováva správaniam elementárne funkcie. Pre správny výber správania sa používa spoločné úložisko údajov, ktoré reprezentuje aktuálny stav sveta. Na základe týchto informácii sa hráč rozhoduje o akcii, ktorú má vykonať.

Správanie bolo rozdelené do viacerých malých modulov kombinovaním ktorých sa ďalej vytvára vyššie a zložitejšie správanie. Jednotliví hráči sa vyznačujú buď obranným alebo útočným správaním. Závisí to od faktu, či daný hráč loptu má, alebo nie. Hráči teda už počas celej hry nemajú napevno daný len jeden určitý typ správania a každý jeden hráč môže byť ako obranca, tak aj útočník.

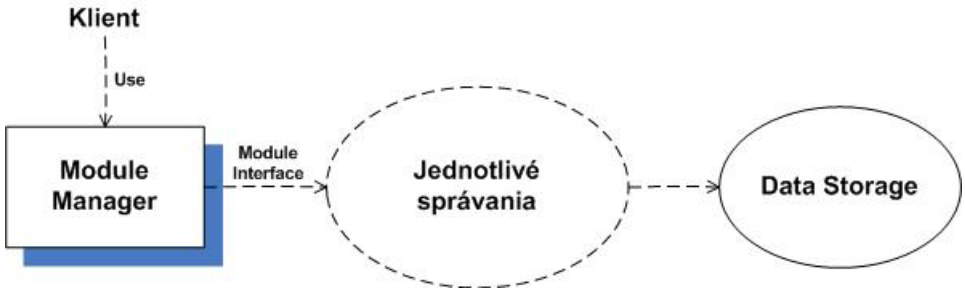
Taktiež bol prerobený aj systém komunikácie agenta so serverom. Bola implementovaná dátová štruktúra, ktorá udržiava informácie o okolitom svete. Túto štruktúru využívajú jednotlivé moduly správania. Na jej základe sa rozhoduje napríklad o vykonaní konkrétnej akcie.

2.1.1.1 Architektúra

Architektúru tímu UTTP znázorňuje obrázok 1 Skladá sa z troch hlavných častí:

- Uchovávanie informácií (trieda *DataStorage*)
- Rozhranie pre prístup k modulom (trieda *ModuleManager*)
- Moduly správaní

Na tomto obrázku je znázornené, ako jednotlivé časti hráča od seba závisia. Trieda Module *Manager* pracuje s jednotlivými typmi správaní prostredníctvom rozhrania *Module Interface*. Jednotlivé druhy správania využívajú informácie uložené v triede *Data Storage*.



Obrázok 1 Architektúra tímu UTTP

Nasleduje podrobnejší opis hlavných častí architektúry tímu UTTP:

- **Uchovávanie informácií** - Trieda *DataStorage* predstavuje centralizovanú správu a úložisko informácií. V tejto štruktúre sa uchovávajú všetky potrebné informácie, ktoré k svojej činnosti potrebujú jednotlivé moduly správaní. Jedná sa konkrétne o modul *Module Manager* a jednotlivé moduly správania. Trieda *DataStorage* je *Singleton* a teda existuje iba jedná jej inštancia, ktorá sa sprístupňuje pomocou makra *USE_MODULE*.

- **Rozhranie pre prístup k modulom** - Trieda *ModuleManager* poskytuje správu a rozhranie pre používanie jednotlivých modulov správaní. Každý modul správania musí byť registrovaný prostredníctvom tejto triedy. Následne po registrácii sa tento modul uloží do mapy modulov správaní.

- **Moduly správaní** - Všetky moduly spolu predstavujú celkové správanie sa tímu. Každý jeden z týchto modulov pokrýva jedno elementárne správanie sa hráča. Napríklad sa môže jednať o prihrávku, odkopnutie lopty alebo driblovanie s loptou. Každý tento modul správania má predpísané spoločné rozhranie.

2.1.1.2 Analýza zdrojového kódu

Hráč tímu UTTP sa vyznačuje multiplatformosťou, je teda kompilovateľný pod systémom *Windows*, ako aj pod systémom *Linux*. Významným prínosom bolo prerobenie zdrojových kódov, aby sa umožnila ich kompilácia v prostredí *MS Visual Studio 2005*. Kompilácia hráča bola otestovaná v prostredí MS *Visual Studia 2005* a následne bol hráč začlenený do hry.

Podarilo sa vylepšiť modularitu kódu. *Module Manager*, ako jedno z hlavných rozšírení má jasnú a prehľadnú štruktúru. Samotný kód tohto modulu je celkom dobre okomentovaný. K dispozícii sú opisy funkcií, na druhej strane však často chýba komentovanie ich tela.

2.1.2 Tím Jahodoví princovia

Hráč tímu Jahodoví princovia vychádza z hráčov tímu *Gang Of Six*, ktorý pôvodne vychádzali zo zdrojových súborov tímu *Uva Trilearn*. Tím *Jahodoví princovia* sa zúčastnil na RoboCup turnaji

2008 v rámci Slovenskej technickej univerzity. Podarilo sa mu dosiahnuť výborné prvé miesto z celkového počtu piatich miest.

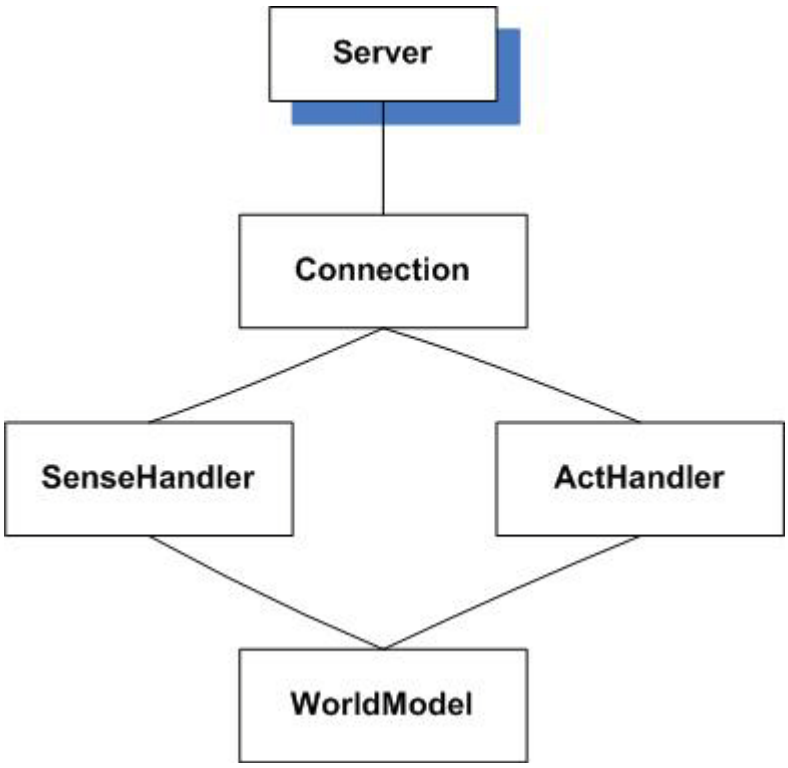
Tím *Jahodoví princovia* sa predovšetkým snažil o rozšírenie hráča o nové vlastnosti. Úprava existujúcich zdrojových kódov sa týkala iba drobných opráv, pričom architektúra hráča ostala zachovaná. Dôvodom boli aj jeho dobré výsledky z turnajov tímov, z ktorých tím *Jahodoví princovia* vychádzal. Rozšírenie hráča sa teda týkalo hlavne nasledujúceho:

- **Vylepšenie vlastností brankára** - snaha o vylepšenie rozohrávania brankára, ktorý niekedy prihrával loptu aj do miest blízko nepriateľa. Po implementácii vylepšenia však brankár občas prihral loptu aj nepriateľovi, pričom tento nedostatok autori nevedeli odstrániť.
- **Zmena formácií počas hry** - tieto zmeny boli založené na základe hodnoty staminy hráčov a pozície lopty na ihrisku. Napríklad, ak by nepriateľ útočil na bránu, všetci hráči by utvorili obrannú formáciu.
- **Algoritmus eliminácie premenných** – cieľom tohto algoritmu bolo vrátiť spoločnú akciu pre tím, ktorá bude maximalizovať spoločnú výnosovú funkciu. V rámci tohto algoritmu boli navrhnuté výnosové funkcie.
- **Výnosové funkcie** - reprezentujú situácie v hre, akými sú napríklad zastavenie súperovej akcie, odkopnutie lopty, streľba na bránu . Tento algoritmus je vlastne stratégiou celého tímu.

Tomuto tímu sa nakoniec podarilo implementovať len veľmi málo z ich plánovaných vylepšení. Neboli implementované všetky výnosové funkcie. Dokonca tieto výnosové funkcie nepriniesli ani žiaden veľký prínos k vylepšeniu hráča. Všetky tieto taktiky už totiž boli v rámci predchádzajúcich fakultných hráčov riešené.

2.1.2.1 Architektúra

Ako už bolo vyššie povedané, architektúra hráča *Jahodoví princovia* ostala nezmenená a vychádza z architektúry hráča *Uva Trilearn*. Túto architektúru naznačuje obrázok 2.



Obrázok 2 Architektúra hráča Jahodoví Princovia

Na najnižšej úrovni pracuje trieda *Connection*, ktorá zabezpečuje spojenie so soccer serverom. O prichádzajúce správy zo strany servera sa stará trieda *SenseHandler*. Naopak, o zaslané správy na server sa zase stará trieda Act*Handler*.

Trieda *WorldModel* predstavuje v tomto tíme najdôležitejšiu časť. Jedná sa konkrétne o model sveta, ktorý obsahuje informácie o aktuálnom stave prostredia hry. Tento model obsahuje aj pravdepodobnostné odhady stavu sveta. K týmto odhadom môžu hráči pristupovať pomocou takzvaných prístupových módov. V rámci tejto triedy existuje aj veľa rôznych metód a atribútov, ktoré zohrávajú dôležitú úlohu pri určovaní akcií, ktoré majú hráči vykonať.

Okrem týchto základných tried tvoria architektúru aj triedy *BasicPlayer* a *Player*. Tieto triedy obsahujú informácie potrebné pre samotné vykonávanie akcií.

2.1.2.2 Analýza zdrojového kódu

Zdrojový kód tímu *Jahodoví Princovia* je veľmi robustný a dobre okomentovaný. Komentáre sú po anglicky, keďže hráč vychádza z tímu *Uva Trilearn*, ktorého dokumentácia je veľmi presná a podrobná. Počas prezerania dokumentácie tohto tímu som však nenašiel žiadne informácie o miestach v kóde, ktoré boli modifikované prípadne doplnené tímom Jahodoví Princovia.

2.1.2.3 Tréner

V rámci projektu vytvoril tím *Jahodoví Princovia* trénera pre testovanie správania vytvorených hráčov a ich zdokonaľovania. Tréner má veľmi jednoduché ovládanie. Prostredníctvom neho dokážeme nastaviť základné vlastnosti akcie ako sú:

- Počet hráčov
- Pozícia hráčov
- Pozícia lopty

2.1.2.4 Systém logovania

V rámci projektu vytvoril tím *Jahodoví Princovia* logovací systém ktorý umožňuje logovanie situácií na ihrisku. V tomto prípade sa zapisujú pozície hráčov do logovacieho súboru.

Po každom odohranom zápase vytvorí server záznam o zápase – tzv. logovací súbor. Prostredníctvom nástroja *Logalyzer* je možné daný textový súbor čítať v príjemnom používateľskom rozhraní. Najzaujímavejšie funkcie tohto nástroja sú:

- Analyzátor dát
- Sledovanie zápasu

Prostredníctvom týchto funkcií môžeme sledovať priebeh zápasu alebo sledovať a kombinovať získané dáta v tabuľkách.

2.1.3 Zhodnotenie výberu tímu, ktorý budeme vylepšovať

Po dôkladnej analýze sme sa rozhodli pre vylepšenie hráča tímu *UTTP*. Je to najmä z dôvodu vysokej úrovne modularity, ktorá bola v tomto hráčovi vytvorená. Jej využitím je možné veľmi jednoduchou cestou dodefinovať, prípadne vylepšiť, správanie sa hráčov. Našou prvoradou snahou teda nie je vytvorenie čo najsilnejšieho hráča, ale skôr toho najkomplexnejšieho, ktorý by bol ľahko rozšíriteľný.

Aj napriek tomu, že tím *UTTP* v rámci turnaja RoboCup neobstál úspešne, sme presvedčení, že jeho rozšírením by bolo možné tento tím posunúť na oveľa vyššiu úroveň.

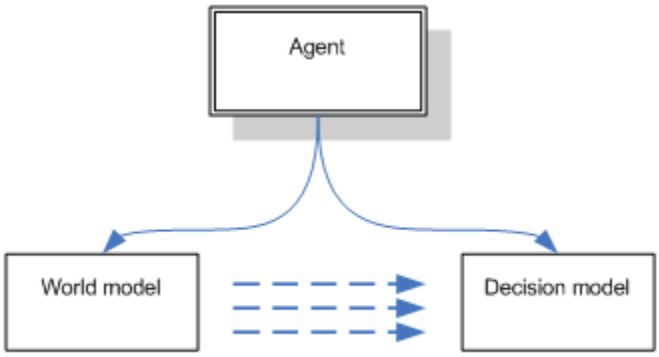
Pri implementácii zmien uvažujeme aj s využitím trénera a systému logovania, ktoré vytvoril tím *Jahodoví Princovia*. Do úvahy taktiež prichádza aj prebratie jednotlivých herných vlastností tohto silného tímu.

2.2 Analýza tímov pre výber vlastností

V tejto kapitole opíšeme tímy, z ktorých neskôr budeme vyberať vlastnosti pre vylepšenie hráča.

2.2.1 Brainstormers

Brainstormers sa zaoberajú témou RoboCup od 1998. Môžu sa pochváliť mnohými úspechmi. Jeden z nich je majstrovský titul z Osaky 2005, potom 2007 a 2008. Ich agent sa dá rozdeliť do dvoch modulov: model sveta a rozhodovací model ako znázorňuje obrázok 3. Model sveta sa skladá z informácií o okolitom svete ako poloha hráča, poloha lopty a podobne. Rozhodovací model rozhoduje o akciách agenta. Vstupom do rozhodovacieho modelu je výstup zo svetového modelu, na základe, ktorého sa vyhodnotí a vyberie optimálna akcia hráča.



Obrázok 3 Brainstormers- rozdelenie agenta na moduly

Rozhodovanie je modelované na základe Markovského rozhodovacieho procesu. Na konečný markovský rozhodovací proces je vhodné použiť učenie s odmenou a trestom (reinforcement learning, RL), čo spravil aj tento tím. Tento typ učenia sa učí bez dohľadu, je založený na vyváženosti využívania už naučených znalostí a skúmania neohodnotených stavov.

Model sveta sa skladá:

- Sebalokalizácia cez časticové filtre
- Pamäť pre uchovávanie stavov a štatistiky
- Komunikácia so serverom
- Ošetrenie parametrov servera

Zručnosti agenta:

- Tri spôsoby driblovania
- Sledovanie a hľadanie lopty
- Pohyb dopredu aj dozadu na špecifikovanú pozíciu
- Štyri spôsoby prebratia lopty
- Tri spôsoby kopnutia do lopty
- Držanie lopty
- Strela na bránku

- Prihrávka a predkopávanie lopty
- Iné

Naučené správanie cez RL:

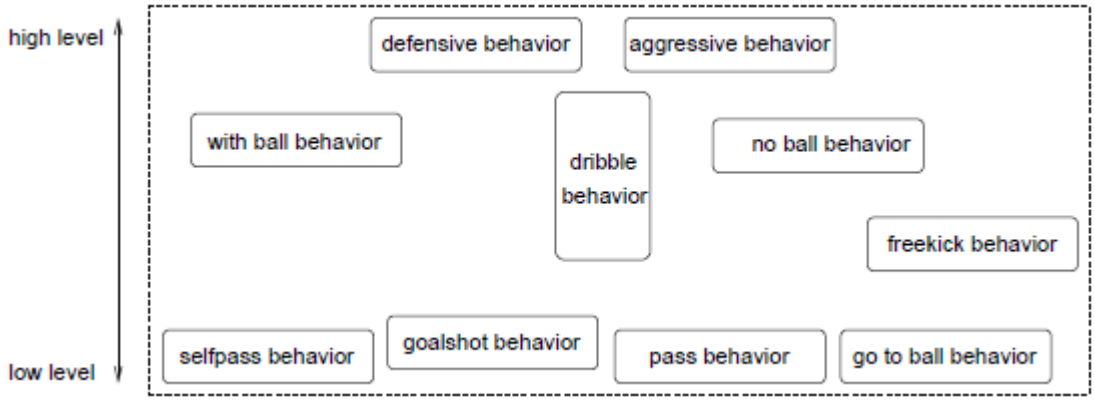
- Správanie na ovládanie krku
- Správanie na ovládanie pohľadu
- Správanie na ovládanie pozornosti resp. pozornosti k čomu

Vďaka RL sú k dispozícii naučené tieto schopnosti: NeuroKick, NeuroKick05, NeuroIntercept, NeuroGo2Pos.

Stredná úroveň schopností: NeuroWball, LearnWball.

Vysoká úroveň schopností: Score04 alebo NeuroPositioning.

Spôsob ako sa hráč môže správať znázorňuje obrázok 4. Hráčovo správanie sa delí na komplexné a jednoduché. Zaujímavé je, že správanie ktoré má na starosti obratie cudzieho hráča sa môže za určitých okolností rozhodnúť neoberať hráča s loptou, ale viac sa sústrediť na defenzívne správanie.



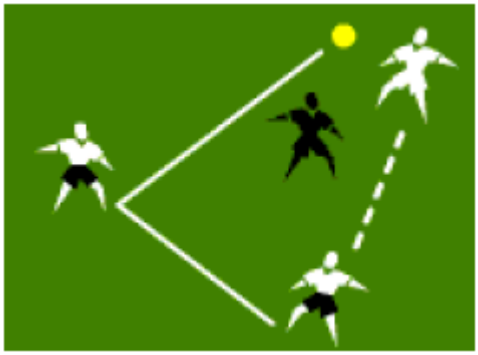
Obrázok 4 Brainstormers - Hierarchia správania[1]

Hráč bol vyvinutý nato, aby sa na ňom mohli skúšať nové stratégie, preto je hráč modulárny. Teoreticky je možné ľahko odoberať a pridávať nové moduly správania.

Zdrojový kód je nerovnomerne okomentovaný, niekedy nie sú okomentované celé bloky kódu.

2.2.2 Oxsy

Tento tím vznikol ako diplomová práca jedného študenta v Rumunsku a tak sa mu to zapáčilo, že sa futbalu venoval aj po skončení štúdia. Tento tím ma implementované zaujímavé myšlienky. Agenti by mali reagovať alebo tvoriť hru ešte predtým ako sa podnet stane. Najlepší príklad je „give-and-go“. Hráč nahrá svojmu voľnému spoluhráčovi a potom začne utekať do voľného priestoru a bude čakať prihrávku od tohto spoluhráča, ktorému nahral prvý krát. V útočnej fáze je dôležité, aby hráč ktorý nahráva predsunutému hráčovi dával pozor na ofsajd.

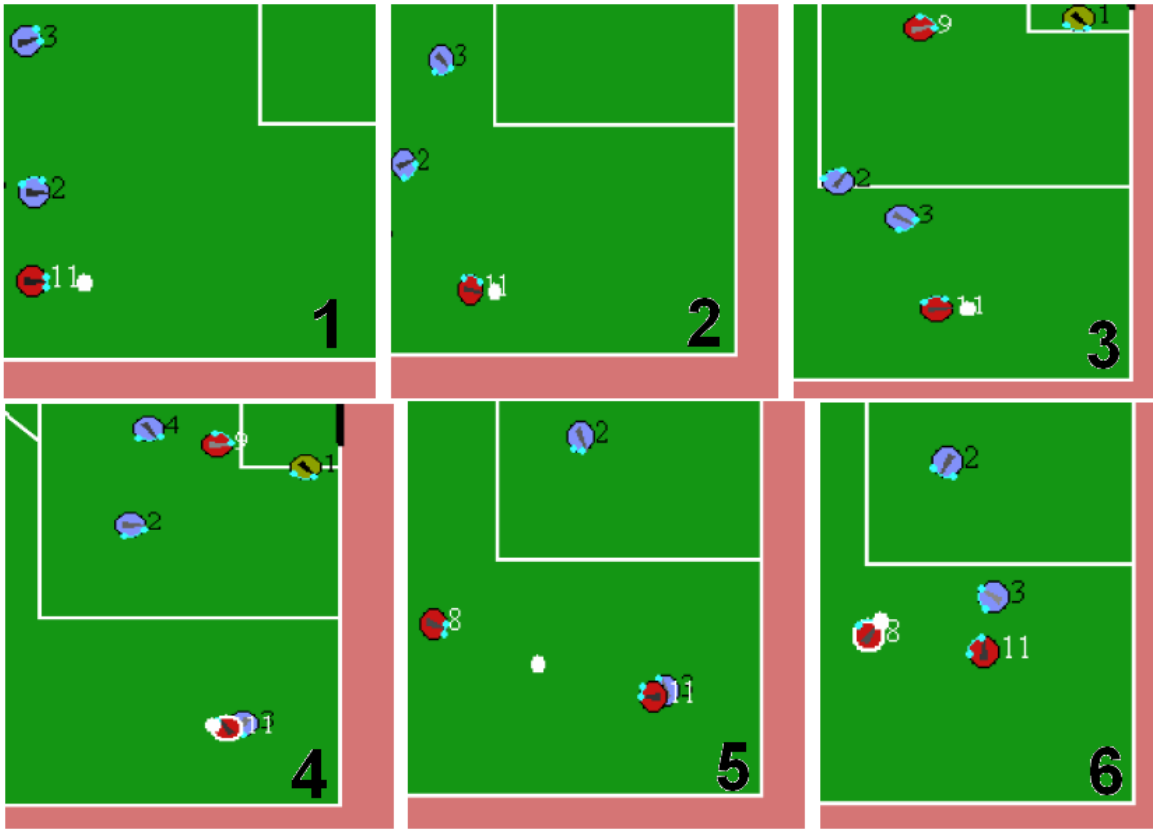


Obrázok 5 Oxsy - "give-and-go"[2]

Ďalšie zaujímavé riešenie sa volá „switch-roles“. Používa sa hlavne v defenzíve. Keď unikajúci hráč s loptou prejde cez obrancu, či už ho prešprintuje alebo ho obkľučkuje, tak hráč stojaci v blízkej vzdialenosti preberie jeho rolu a ide sa snažiť tomuto prenikajúcemu útočníkovi vypichnúť loptu. Tak pôvodný hráč, ktorého obkľučkoval si zoberie úlohu tohto obrancu, ktorý práve ide po útočníkovi.

Tento tím sa už údajne ďalej nevyvíja, lebo jeho autor prechádza na 3D RoboCup aspoň to tak avizoval, no na súťažiach sa Oxsy stále objavuje, takže vývoj asi napriek poplašným správam

prebieha naďalej. Zdrojový kód tohto tímu, žiaľ nie je prístupný. Sprístupnená časť zdrojového kódu tvorila len oklieštené primitívne funkcie, bez žiadnych techník.



Obrázok 6 Oxsy - “Switch roles” [2]

2.2.3 Tsinghuaeolus

Tsinghuaeolus [7] je tím, ktorý vznikol v roku 2000 v Pekingu na univerzite Tsinghua. Prístupy k samotnej logike hráča sú rozdelené podľa úrovní. Na úrovni základných zručností používa tím techniky, ktoré sú založené na učení. Na vyššej, strategickej úrovni používa analytické prístupy, ktoré sú ľahšie implementovateľné a efektívnejšie. Dostupné sú zdrojové kódy z roku 2002. Sú síce staršie, ale stoja za povšimnutie. Na nižšej úrovni sa sústredili najmä na situácie, keď je hráč ohrozovaný hráčmi súpera.

Medzi základné zručnosti patria kopanie do lopty a driblovanie v nepriateľskom prostredí.

Driblovanie v nepriateľskom prostredí je realizované tak, že pri každom cykle je udržiavaná lopta na ploche, v ktorej sa dá do nej kopať. Avšak hráč musí brať tiež na zreteľ, aby udržiaval loptu mimo plôch, v ktorých môže kopať súper. Hráč sa musí najprv uistiť, či môže zmeniť smer lopty v prípade, že sa blíži súper. Až potom sa môže pohnúť dopredu.

Pri kopaní do lopty v nepriateľskom prostredí je okrem udržania lopty mimo dosahu súpera, nutné dávať pozor aj na postranné čiary. Hráč, aby kopol loptu želaným smerom, musí spraviť niekoľko prípravných krokov. Musí si loptu pritiahnuť a na základe jej rýchlosti a smeru, určiť na čo najmenší počet prípravných kopnutí želaný postup. Toto realizuje pomocou algoritmu strojového učenia sa.

2.2.4 UvA Trilearn 2003

Tím UvA Trilearn patrí medzi najúspešnejšie tímy, v roku 2001 a 2002 získali 2. miesto. Dostupné zdrojové kódy sú kvalitné a dostatočne okomentované, ale z pochopiteľných dôvodov nezverejnili všetky časti kódu. Týka sa to najmä rozhodovacieho stromu a koordinačných grafov použitých pri riadení agentov.

Správanie hráčov (okrem brankára) je modelované pomocou rozhodovacieho stromu. V prvom kroku sa hráč snaží zistiť pozíciu lopty. V závislosti na vzdialenosti hráča od lopty sa vykoná jedna z nasledovných akcií:

1. ak sa lopta nachádza v blízkosti hráča, tak hráč kopne do lopty najväčšou možnou silou, pričom kopnutie smeruje do náhodne vybraného rohu brány protihráča.

2. keď loptu nemá nikto, potom najrýchlejší hráč (hráč, ktorý potrebuje najmenej cyklov na beh k lopte) sa snaží získať loptu. Ak ju získa, tak do nej kopne.

3. ostatní hráči sa usilujú o získanie najvýhodnejšej pozície na ihrisku v závislosti od svojej pozície a pozície lopty.

Pri rozhodovaní sa zohľadňuje aj fitnes hráčov, teda hráč so slabou vitalitou vynaloží menej úsilia pri vykonaní akcie ako hráč so silnejšou vitalitou. Správanie brankára je tiež určené rozhodovacím stromom, a to nasledovným spôsobom. Keď sa lopta dostane do pokutového územia, predpovedá sa trajektória lopty pomocou úsečky. Krajné body úsečky sú pozícia lopty a stred brány. Ak okolie brankára, v ktorom dokáže chytiť loptu, obsahuje časť tejto úsečky, potom bankár chytí loptu. V opačnom prípade sa postupne približuje k čiare a ku svojej bráne.

2.2.4.1 Správanie hráča na nižšej úrovni

Hľadanie lopty

Keď hráč nevidí loptu, najprv sa pozrie do svojho modelu sveta, kde má poznačenú poslednú pozíciu lopty. Potom sa postupne otáča smerom k predchádzajúcej pozícii lopty tak, aby snímal neskúmanú časť ihriska.

Spracovanie lopty

Pred spracovaním pohybujúcej sa lopty ju musíme zastaviť. Keďže pohyb lopty je implementovaný ako sčítanie vektorov, hráč musí kopnúť do lopty takým spôsobom, aby generoval vektor, ktorý má takú istú veľkosť ako vektor lopty ale opačný smer. Najväčšia efektivita sa dosahuje vtedy, ak je hráč otočený smerom k prichádzajúcej lopte.

Kopnutie lopty

Pred kopnutím sa vypočíta sila potrebná k tomu, aby sa lopta dostala na želanú pozíciu. Ak táto sila je menšia než maximálna sila ktorou môžeme kopnúť, kopnutie lopty sa realizuje v jednom cykle. V opačnom prípade sa hráč pohybuje s loptou smerom k jej cieľovej pozícii dovtedy, kým ju nedokáže posunúť do cieľovej polohy jediným kopnutím.

2.2.4.2 Správanie hráča na vyššej úrovni

Získanie lopty

Pri tomto správaní sa využíva predikcia pozície lopty v nasledujúcich dvoch cykloch. Najprv sa vypočíta poloha lopty v nasledujúcom cykle a zistí sa, či hráč dokáže získať loptu posunutím sa do predpovedanej pozície lopty bez otáčania sa. Ak áno, potom hráč získa loptu v nasledujúcom cykle. V opačnom prípade sa predpovedá pozícia lopty o 2 cykly neskôr, v 1. kroku sa hráč posunie, v 2. sa otočí takým smerom, aby mohol získať (spracovať) loptu. A hráč nemá dostatok energie na to, aby sa posunul na želanú pozíciu, potom možnosť získania lopty sa vyhodnotí záporne.

Driblovanie

Hráč UvA Trilearn používa nasledovné tri typy driblovania:

- rýchle – kopnutie lopty ďaleko od seba
- pomalé – kopnutie lopty na menšiu vzdialenosť od seba
- s loptou – držanie lopty veľmi blízko seba

Driblovanie sa uskutoční, ak hráč dokáže manipulovať s loptou (lopta sa nachádza v jeho kickable_area) a zároveň má loptu pred sebou. Driblovanie je úzko spojené so získavaním lopty: po kopnutí treba loptu získať späť aby sme mohli pokračovať v driblovaní.

Prihrávanie

Pri prihrávaní sa využívajú nasledovné 3 prístupy:

- direct – cieľová pozícia lopty je poloha spoluhráča
- leading – prihrávanie lopty do otvoreného priestoru, kam smeruje aj spoluhráč, ktorý ju dokáže spracovať v jednom cykle (teda nemusí sa otáčať)
- through – prihrávanie do otvoreného priestoru medzi brankárom a obrancami opačného tímu tak, aby spoluhráč bol schopný získať loptu skôr než protihráči.

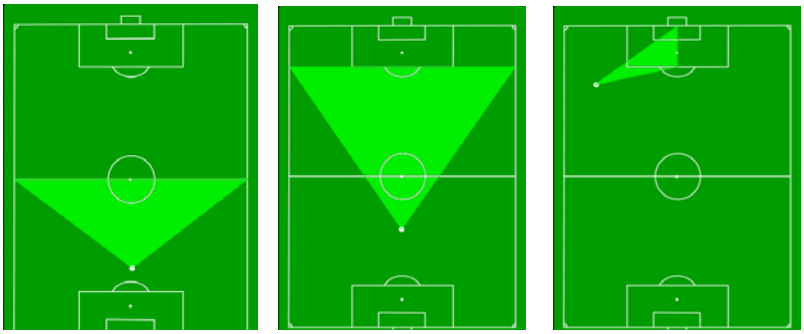
Situácia útočník – obranca

Útočník, ktorý má pri sebe loptu a pred sebou obrancu, sa nachádza v pozičnej výhode. Keďže pohyb s loptou je pomalší než bez nej, útočník kopne loptu na také miesto na ihrisku, kam sa dokáže presunúť skôr než obranca. Útočník má výhodu v tom, že po kopnutí lopty sa už nemusí otáčať smerom k lopte, kým obranca áno. Tým pádom sa k lopte dostane o cyklus skôr než protihráč.

Uvoľnenie

Ak obranca v nebezpečnej situácii nemôže driblovať alebo prihrať loptu spoluhráčovi s najväčšou silou, kopne loptu do vopred určenej oblasti ihriska. Táto oblasť sa definuje na základe typu uvoľnenia takto:

- clear ball defensive – trojuholníková oblasť definovaná pozíciou lopty a stredovou čiarou ihriska
- clear ball offensice - trojuholníková oblasť definovaná pozíciou lopty a čiarou, ktorá obsahuje čiaru pokutového územia a jej koncové body sa nachádzajú na postrannej čiare hracej plochy
- clear ball goal - trojuholníková oblasť definovaná pozíciou lopty a stredom brány



Obrázok 7 Uvoľnenie: clear ball defensive, clear ball offensive, clear ball goal

Sledovanie hráča

Toto správanie umožňuje strážiť protihráča, teda nedovoliť, aby sa k nemu dostala lopta. Rozlišujeme 3 typy sledovania:

- mark ball – obranca stojí medzi loptou a protihráčom, zabráni prihrávke
- mark goal – obranca stojí medzi protihráčom a stredovým bodom svojej bránkovej čiary, a tak zabráni gólu
- mark bisector – obranca sa nachádza v trojuholníkovej oblasti, ktorá je definovaná pozíciou lopty, pozíciou protihráča a stredom brány. Týmto spôsobom sledovania dokáže hráč blokovať priame (direct) aj nepriame (leading) prihrávky.

2.2.5 DSL United (2003)

Projekt DSL United vznikol ako súčasť výskumného projektu zaoberajúceho sa rôznymi systémami umelej inteligencie. Zdrojové kódy, ktoré boli dostupné na stránke projektu, neboli okomentované v dostatočnej miere a navyše sa vyskytol problém pri kompilácii projektu, čo v značnej miere obmedzuje možnosť použitia vybraných častí kódu. Rozhodovací strom správania sa hráča je pomerne jednoduchý a realizuje sa nasledovným spôsobom:

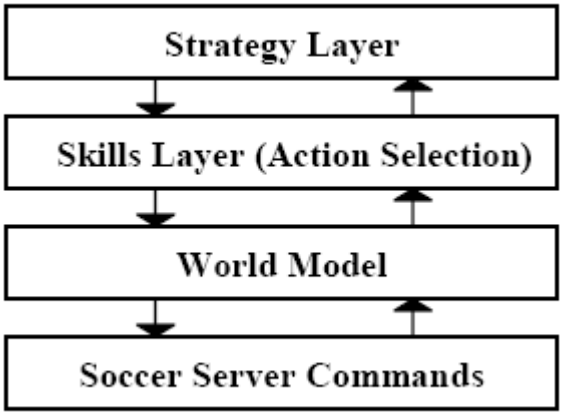
- Ak má hráč loptu a jeho protihráč je ďaleko, používa rýchle driblovanie.
- Ak má hráč loptu a jeho protihráč je blízko, potom
  - ak spoluhráč je blízko, prihrá
  - ak spoluhráč je ďaleko ale je blízko k bráne protihráča, strelí na bránu
  - v opačnom prípade používa pomalé driblovanie.

- Ak hráč nemá loptu a jeho spoluhráč je bližšie k lopte ako on, potom sa posunie do defenzívy, v opačnom prípade beží k lopte.

- Defenzíva – pri defenzíve sa hráč obzrie okolo seba, identifikuje loptu a najbližšieho protihráča, ktorý ju môže dostať. Potom sa posunie do takej pozície aby bol medzi loptou a týmto protihráčom.

2.2.6 Nexus2D

Nexus2D [9] je hráčom vytvoreným Iránskym tímom na Ferdowsi University of Mashhad. Jeho architektúra je založená na princípe viacerých vrstiev, pri ktorých každá vrstva dokáže spracovať informácie, ktoré jej poskytla vrstva, ktorá je o úroveň nižšie. Taktiež samozrejme poskytuje svoj výstup vrstve, ktorá je o úroveň vyššie. Bližší model je vidieť na obrázku 8:

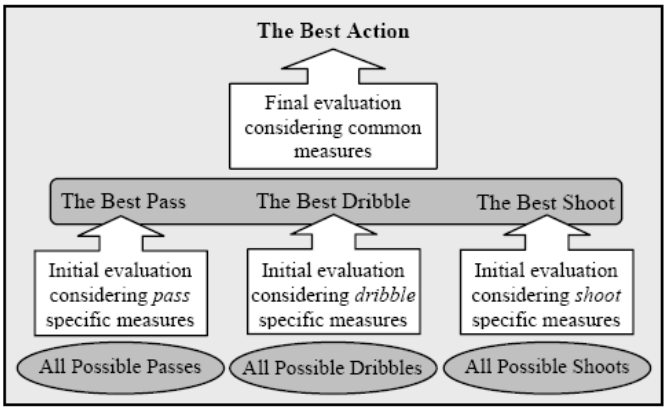


Obrázok 8 Vrstvový model hráča Nexus2D [9]

Strategická vrstva, je vrstva, ktorá využíva informácie z vnútorného modelu sveta a základné schopnosti pre hranie hry.

Hráč Nexus2D pracuje na princípe dvojfázového rozhodovacieho mechanizmu. Pri rozhodovaní sa hráča, akú akciu má uskutočniť najskôr, vyberie sa zvlášť najlepšia možná strela, najlepšia

možná prihrávka a najlepšia možná cesta pre driblovanie (pohyb s loptou) a v druhom kroku sa vyberie najlepšia možná akcia z predchádzajúcich troch najlepších akcií. Výber najlepších akcií je ovplyvnený váhami, ktoré sa po každom výbere najlepšej akcie menia. Podrobnejšie opisuje tento model obrázok 9.



Obrázok 9 Dvojfázové rozhodovanie sa hráča [9]

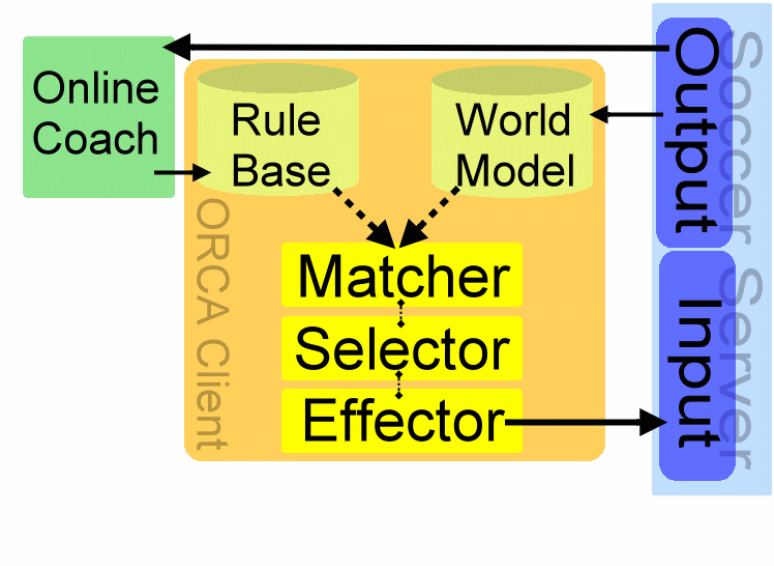
2.2.7 Dirty Dozen

Dirty Dozen [8] je hráčom, vytvoreným nemeckým tímom ORCA (Osnabrueck RoboCup Agents). Je postavený na kóde CMU (Carnegie Mellon University) z roku 1999.

Hráč Dirty Dozen pracuje na princípe rozhodovacích stromov, každá informácia z vonkajšieho sveta sa ukladá do vnútornej reprezentácie sveta. Práve na základe rozhodovacích stromov a vnútornej reprezentácie sveta sa rozhodne o nasledujúcej akcii. Okrem vnútorného modelu sveta má k dispozícii hráč taktiež informácie od trénera, ktoré si ukladá do bázy znalostí. Tieto informácie už len rozširujú vopred naplnenú bázu znalostí s vopred určenou stratégiou hry. Hráč využíva Strategy Formalization Language (SFL), čiže rozšírenie jazyka CLang.

Tento systém reprezentujú tri komponenty: Matcher, Selector a Effector. Prvý z nich porovnáva informácie z vnútornej reprezentácie sveta s informáciami uloženými v znalostnej báze. Ak nájde

viacero možností, spomedzi ktorých sa možno rozhodnúť, toto rozhodnutie uskutoční Selector. Vyberá sa pravdaže najvhodnejšie pravidlo. Samotnú realizáciu pravidla uskutoční Effector.



Obrázok 10 Vnútorný model hráča DirtyDozen [8]

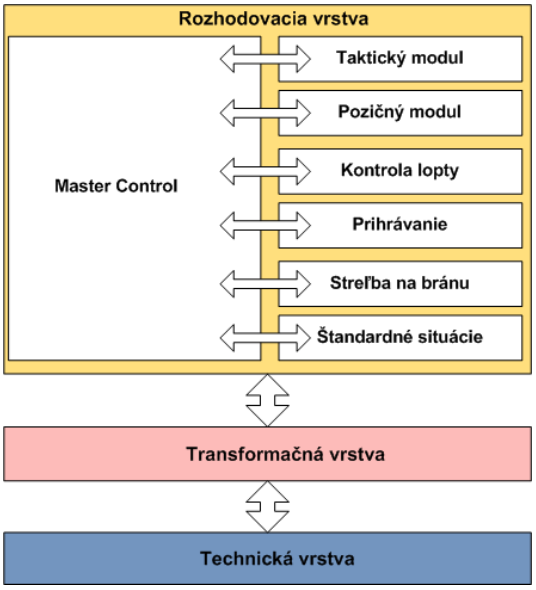
Tento hráč počúva vlastného kouča, ktorý bol implementovaný pre účely a stratégie tohto konkrétneho tímu.

2.2.8 Mainz Rolling Brains

Mainz Rolling Brains [3] je tím pochádzajúci z Nemecka. Analyzované zdrojové kódy sú z roku 2002. Implementácia agenta je rozdelená do troch vrstiev (pozri obrázok 11) a ich význam je nasledovný:

1. *Technická vrstva* - zabezpečuje komunikáciu so serverom, t. j. odosielanie a spracovávanie príkazov podľa dohodnutého komunikačného protokolu,
2. *Transformačná vrstva* - spracovávajú a uchovávajú sa tu údaje o prostredí (model prostredia), a taktiež sa realizujú pomocou tejto vrstvy niektoré z rozšírených schopností agenta (napríklad kopnutie do lopty na určenú pozíciu a pod.),

3. *Rozhodovacia vrstva* - takpovediac „mozog" hráča. Pozostáva z niekoľkých menších modulov, ktoré sú určené na špecifickú oblasť správania sa hráča. Úlohou týchto modulov je nielen realizácia prislúchajúcich akcií, ale i ohodnotenie vhodnosti ich vykonania v konkrétnej hracej situácii. Na ohodnotenie pritom moduly používajú šesť diskrétnych stupňov (od „nemožné" až po „absolútne"). Špeciálny význam v tejto vrstve má modul *Master Control*, ktorý slúži na výber akcie, ktorá sa skutočne realizuje v danej situácii. Rozhoduje sa pritom podľa ohodnotení, ktoré získal z jednotlivých modulov. Taktický modul napríklad uchováva údaje o role hráča v tíme (brankár, obranca, stredopoliar, útočník), údaj o aktuálnom držaní lopty (oponent alebo hráč nášho tímu) atď. Význam ostatných modulov je zrejmý z ich slovného označenia.



Obrázok 11 Trojvrstvový model implementácie agenta [3]

Model prostredia obsahuje nielen aktuálny stav ale i predpokladaný stav v budúcnosti. Ten sa určuje extrapoláciou aktuálnych (známych) údajov a znalosti dynamiky (napr. pozícia a rýchlosť) lopty a ostatných hráčov.

Veľmi zaujímavou vlastnosťou, s ktorou sme sa stretli pri analýze tohto tímu, je komunikácia medzi hráčmi pomocou krátkych správ. Správy pritom môžeme podľa ich významu rozdeliť do dvoch kategórií:

1. *Strategické ciele* - úlohou týchto správ je kvalitatívne zlepšenie hry, t. j. aby za loptou nevyrazilo naraz niekoľko hráčov a pod.,
2. *Informácie o objektoch v hracom poli* - správy slúžia na spresnenie údajov o aktuálnom stave sveta.

Ak sa hráč rozhodne uskutočniť prihrávku, informuje o svojom zámere svoje okolie, t. j. spoluhráčov nachádzajúcich sa v jeho blízkom okolí. Ak tí vyhodnotia svoju situáciu ako výhodnú (hráčova stamina je na vysokej úrovni a nachádza sa v blízkosti súperovej bránky, protihráči sú od neho dostatočne ďaleko atď.) môžu požadovať prihrávku ich smerom. Naopak, ak hráč vyhodnotí svoju situáciu ako nevýhodnú (má nízku stamina a v jeho blízkosti sa nachádza protihráč atď.), informuje spoluhráčov o tom, aby na neho nerealizovali prihrávku. Obdobne sa hráči informujú o tom, kto smeruje k lopte so zámerom ju zachytiť. Takýmto spôsobom sa eliminuje riziko, aby za loptou zbytočne nevyšlo viacero hráčov. Pre tímovú hru sú správy, ktoré si medzi sebou vymieňajú spoluhráči, veľmi cenné. Treba však poznamenať, že v žiadnom prípade nie sú nevyhnutnosťou pre rozhodovanie agenta.

Schopnosti hráčov sú rozdelené do dvoch úrovní: základné a rozšírené. Medzi základné patria tie, ktoré sú priamo posielané na server (napr. kopni do lopty). Komplexnejšie zručnosti sú kombináciou viacerých základných (napr. driblovanie). Aktivita hráča, ktorý nekontroluje loptu, je daná jeho úlohou v tíme a taktiež vyplýva z aktuálnej situácie. Ak sa teda nachádza blízko pri lopte, je pravdepodobné že pôjde jej smerom s cieľom získať nad ňou kontrolu, inak sa bude snažiť zaujať optimálne postavenie (napr. s možnosťou výhodnej prihrávky a pod.).

Zdrojové kódy sú zrozumiteľné a dobre štruktúrované. Názvy tried, premenných a metód vystihujú ich účel a uľahčujú pochopenie jednotlivých častí kódu. Zdrojový kód je navyše veľmi dobre okomentovaný. Základná funkcionalita tvorí knižnicu, ktorú využívajú jednotlivé špecializované moduly. Pre účely nášho projektu by sme mohli použiť jednu z rozšírených

schopností hráča, konkrétne driblovanie. Inšpirovať sa môžeme taktiež myšlienkou komunikácie medzi hráčmi pomocou krátkych správ.

2.2.9 YowAI

Tím YowAI [4] pochádza z Japonska a dosiahol dobré výsledky v rámci regionálnych kôl. Zdrojové kódy pochádzajú z roku 2002. Základnou myšlienkou, ktorú autori zabudovali do svojho projektu, je snaha o napodobnenie správania sa ľudského hráča pri rozhodovaní o vlastných akciách a kooperácii so spoluhráčmi. Tak ako v skutočnom futbale aj tu medzi sebou jednotliví hráči komunikujú. Komunikácia má v tomto prípade formu krátkych hlášok ako napríklad „strieľaj“, „prihraj“ a pod. Zmyslom je poskytovať hráčom rady ako sa majú zachovať v konkrétnej situácii a pomôcť im pri výbere vhodnej akcie. Úspešné použitie tejto metódy však vyžaduje dobré individuálne taktické a strategické schopnosti hráča.

Z dôvodu presnejšieho rozpoznania situácie a pravdepodobných akcií oponenta sú agenti vybavení jednoduchou „pamäťou“, v ktorej sú uchované informácie o čase, pozícii a rýchlosti lopty, pozícii agentov a doplnkové informácie (ktorý agent bol najbližšie, kto kontroloval loptu). Vďaka prítomnosti údajov nielen o aktuálnej pozícii ale aj o krátkej histórii, je možné rozpoznať správanie sa oponenta. Napríklad podľa toho, kto momentálne kontroluje loptu je možné zmeniť herný štýl z ofenzívy na defenzívu a naopak, alebo je možné rozpoznať hráča, ktorý kontroluje loptu (driblovanie).

Zdrojové kódy sú prehľadné a vhodne štruktúrované. Vážnym nedostatkom, s ktorým sme sa stretli pri ich analýze je, že nie sú dostatočne zdokumentované. Aj to málo komentárov, ktoré sme v zdrojových kódoch našli, je prakticky nepoužiteľné. Dôvodom je absencia svetového jazyka. To značne predlžuje dobu potrebnú na pochopenie implementovanej logiky. Aj z tohto dôvodu preto nepokladáme za vhodné použiť žiadnu z častí tohto zdrojového kódu. Za zmienku stojí len myšlienka komunikácie medzi hráčmi, ktorá je však podľa nás lepšie implementovaná v agentovi tímu Mainz Rolling Brains.

2.3 Opis soccer servera

Soccer server je prostredie, prostredníctvom ktorého sa uskutočňujú zápasy medzi jednotlivými tímami. Toto prostredie zároveň zabezpečuje priebeh zápasu prostredníctvom pravidiel, ktoré kontroluje rozhodca. Klienti, teda hráči, komunikujú so serverom prostredníctvom protokolu UDP. Okrem hráčov a rozhodcu existuje aj tretí účastník zápasu a to kouč, ktorý môže prostredníctvom pokynov ovplyvňovať priebeh zápasu. Priebeh zápasu je možné sledovať prostredníctvom aplikácie SoccerMonitor.

2.3.1 Typy komunikácie medzi serverom a klientom

Hráč má možnosť komunikovať so serverom cez nasledovné protokoly:

- Príkazový protokol (command protocol) – zabezpečuje pripájanie, odpájanie na server a ovládanie hráča.
- Vnemový protokol (client sense protocol) - server zasiela hráčovi informácie o virtuálnom zápase – rozmiestnenie hráčov, strategických bodov, informácie o polohe lopty a hráčove stavové informácie.

2.3.2 Informácie získané zo senzorov

Aby mohol hráč interaktívne reagovať na aktuálnu situáciu na ihrisku, je potrebné, aby poznal súčasný stav v jeho okolí a podnety, na ktoré by mal vedieť reagovať. Z týchto dôvodov soccer server poskytuje informácie o dianí na ihrisku prostredníctvom vizuálneho vnemu, sluchového vnemu a telového vnemu.

2.3.2.1 Vizuálny vnem

Tento vnem poskytuje informácie o dianí, ktoré je hráčom aktuálne pozorované. Tieto informácie sú zasielané hráčovi v nasledujúcom formáte:

(see <ObjName> <Distance> <Direction> <DistChng> <DirChng> <FaceDir>)

kde:

- *ObjName* identifikuje pozorovaný objekt

- *Distance, Direction, DistChng, DirChng, FaceDir* sú informácie o relatívnej vzdialenosti, smere, natočení hráča a pod.

Hráč môže vidieť objekty dvoma spôsobmi:

1. zorné pole – hráč rozpoznáva objekty vo svojom zornom poli, tvoreným uhlom od -90° do 90°, resp. iným uhlom v závislosti od typu pohľadu
2. susedstvo – ak je objekt v relatívnej vzdialenosti menšej ako 3 m a zároveň nie je v zornom poli hráča, hráč dokáže identifikovať len typ objektu, bez jeho bližšieho poznania mena

Zvolený pohľad hráča rozhoduje o tom, ako často mu budú informácie o videných objektoch zasielané a v akej kvalite, t.j. koľko informácií o nich dostane. Pohľad je nastaviteľný za pomoci dvojice ANGLE_WIDTH a ANGLE_QUALITY.

ANGLE_WIDTH musí byť jedna z hodnôt

- *wide* = 180° <-90,90>

- *normal* = 90° <-45,45>

- *narrow* = 45° <-22.5,22.5>

ANGLE_QUALITY musí byť

- *low* (je posielaná iba informácia o smere)

- *high* (klient dostane informáciu aj o smere aj o vzdialenosti objektu).

Prednastavená hodnota parametra ANGLE_WIDTH je *normal* a parametra ANGLE_QUALITY je *high*.

2.4 Evolúcia soccer servra od verzie 9.0.4 až po 13.0.0

V tejto časti sú opísané zmeny soccer servera medzi verziami 9.0.4 až 13.0.0. Poslúžia ako zdroj informácií pre upgrade existujúceho hráča, ktorý používal komunikačný protokol pre verziu servera 9.0.4 tak, aby sa dali využiť vlastnosti servera 13.0.0.

2.4.1 Verzia 9.0.4

- Podpora nového režimu KeepAway: tím na ľavej strane ihriska (keepers) sa snaží o udržanie lopty a tím na pravej strane hracej plochy (takers) sa ju snaží získať. Epizóda sa skončí, keď tím na pravej strane získa loptu, alebo keď lopta opustí hraciu plochu. Režim Keepaway spustí použitím parametra keepaway, simulácia sa zapisuje do súboru s príponou .kwy.

- Podpora používania dynamických knižníc; časovače sú načítane takýmto spôsobom na platformách, ktoré to podporujú

2.4.2 Verzia 10.0.0

- Úprava štruktúry adresárov soccer servera

2.4.3 Verzia 11.0.0

- Podpora 64 bitových operačných systémov

- nový voliteľný parameter servera coach_msg_file: definuje súbor ktorého obsah sa pošle trénerovi ako správa zo servera

- automatické pokračovanie v hre: ak režim hry je play_on a lopta sa nachádza v tzv. ball_stuck_area, po určitom čase (drop_ball_time) rozhodca dá pokyn na pokračovanie hry.

2.4.4 Verzia 12.0.0

- Úprava monitorovacieho protokolu, verzia 3 je predvolená
- Úprava formátu logu hry, verzia 3 je predvolená

2.4.5 Verzia 12.1.0

- Zmena formátu logu hry na verziu 4

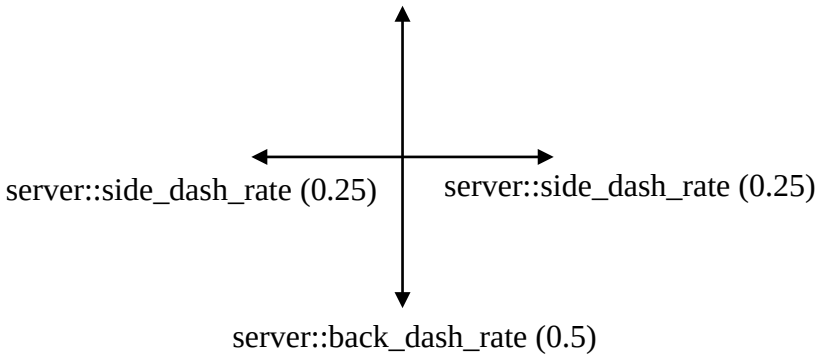
2.4.6 Verzia 13.0.0

- Úprava balíčkov: zdrojové súbory rcsssbase sa stali súčasťou zdrojových súborov rcsserver.
- Nové parametre servera:
    - server::stamina_capacity (predvolená hodnota: 148600.0)
    - server::max_dash_angle (predvolená hodnota: 180.0)
    - server::min_dash_angle (predvolená hodnota: -180.0)
    - server::dash_angle_step (predvolená hodnota: 90.0)
    - server::side_dash_rate (predvolená hodnota: 0.25)
    - server::back_dash_rate (predvolená hodnota: 0.5)
    - server::max_dash_power (predvolená hodnota: 100.0)
    - server::min_dash_power (predvolená hodnota: -100.0)
- Zmenené parametre servera:
    - server::stamina_max (4000 -> 8000)

- server::max_back_tackle_power (50.0 -> 0.0)
- server::extra_half_time (300 -> 100)
- server::player_speed_max (1.2 -> 1.05)
- server::player_speed_max_min (0.8 -> 0.75)
- server::extra_stamina (0.0 -> 50.0)
- player::player_decay_delta_min (-0.05 -> -0.1)
- player::extra_stamina_delta_max (100.0 -> 50.0)
- player::effort_mix_delta_factor (-0.002 -> -0.004)
- player::effort_man_delta_factor (-0.002 -> -0.004)
- player::new_dash_power_rate_delta_min (-0.0005 -> -0.0012)
- player::new_dash_power_rate_delta_max (0.0015 -> 0.0008)

- Zmeny v modeli výdrže (stamina) hráča: parameter server::stamina_capacity definuje maximálne množstvo energie, ktoré sa hráčovi dopĺňa počas zápasu. Keď sa hráčovi dopĺňa energia, hodnota premennej stamina_capacity klesá a keď dosiahne nulu, jeho energia sa neobnoví. Záporná hodnota tejto premennej znamená nekonečnú kapacitu energie a v tomto prípade sa model energie rovná modelu z predchádzajúcich verzií servera.

Zmenený model šprintovania: hráč môže určiť ktorým smerom chce šprintovať: smer sa počíta relatívne od polohy hráča a jeho hodnota je v intervale <server::min_dash_angle,server::max_dash_angle>



Efektivita šprintu tiež závisí od jeho smeru a nastavuje sa pomocou nasledujúcich parametrov:

- pomer k efektivite šprintu smerom dopredu
    - Pomocou parametra server::dash_angle_step je možné určiť uhol šprintu, t.j. ak hodnota tejto premennej je 90, hráč môže šprintovať štyrmi rôznymi smermi.
- Podpora verzie 4 monitorovacieho protokolu a verzie 5 formátu logovania.

3 Špecifikácia

V tejto kapitole sú spomenuté vlastnosti, ktoré požadujeme od výsledného produktu

Naším hlavným cieľom je:

- Zlepšiť už existujúce správanie sa hráča, prípadne doplniť nové prebratím z cudzích tímov
- Implementovať rozhodovanie vo výbere medzi typmi správania sa toho istého druhu. Napríklad kedy je lepšie použiť ktorý spôsob driblovania s loptou
- Zachovať, prípadne ešte zvýšiť modularitu existujúcej architektúry hráča
- Implementovať komunikáciu medzi hráčmi navzájom za účelom zlepšenia herných vlastností počas hry

3.1 Špecifikácia požiadaviek

V nasledujúcich odstavoch sú vymenované požiadavky, ktoré je potrebné pokryť funkcionalitou hráča. Bližšie spresnenie sa nachádza v ďalších podkapitolách.

P1.0 - Multiplatformovosť

Hráč bude kompilovateľný a prevádzkovateľný na prostrediach:

- P1.1 GNU/Linux
- P1.2 MS Windows XP
- P1.3 MS Windows Vista

P2.0 – Modularita

Hráč bude mať funkcionalitu rozdelenú do modulov. V jednom module môže byť len funkcie, ktoré so sebou navzájom bezprostredne súvisia aby sa zabezpečila jednoduchá udržateľnosť kódu.

P3.0 – Využitie základu existujúceho hráča

Ako základ bude využitý už existujúci hráč, vnútorné rozhrania častí hráča sa prispôsobia tak, aby boli všeobecne využiteľné.

P4.0 - Funkcionalita nižšej úrovne

Funkcionalitu nižšej úrovne preberieme z existujúceho hráča UTTP. Rozšírime ju o ďalšiu funkcionalitu prebranú zo zahraničných hráčov, alebo o funkcionalitu, ktorú odznova naimplementujeme.

- P4.1 Kopanie do lopty
- P4.2 Prihrávky
- P4.3 Driblovanie
- P4.4 Strely na bránu
- P4.5 Vykopávanie lopty
- P4.5 Správanie sa hráča bez lopty

P5.0 -Funkcionalita vyššej úrovne

Funkcionalitu vyššej úrovne preberieme z existujúceho hráča UTTP a vylepšíme ju.

- P5.1 Stratégia pri útoku
- P5.2 Stratégia pri obrane

3.2 Špecifikácia vybranej funkcionality

V tomto tímovom projekte ideme rozširovať funkcionalitu existujúceho fakultného hráča (požiadavka P3.0), aby sme nadviazali na minuloročné práce vytvorené na tímových a záverečných projektoch.

Ako základ pre implementáciu sme si vybrali hráča tímu UTTP, pretože spĺňa požiadavku P2.0 a tou je modularita samotného hráča. Modularita uľahčuje orientáciu v zdrojovom kóde a vytvára lepší základ pre doimplementovanie ďalších vlastností. Ak to bude potrebné, upravíme rozhrania existujúceho hráča, aby sme uľahčili implementáciu ďalšej funkcionality.

Hráč by mal byť spustiteľný a kompilovateľný na prostrediach GNU/Linux, MS Windows XP a MS Windows Vista. GNU/Linux bude primárne prostredie určené pre beh hráča, keďže najnovší vývoj Servera a ostatných aplikácií projektu RoboCup prebieha práve na tejto platforme.

Funkcionalitu vyššej úrovne (požiadavka P5.0) preberieme predovšetkým od fakultného hráča, ktorého budeme rozširovať. Niektoré prvky funkcionality vyššej úrovne doimplementujeme.

Funkcionalitu nižšej úrovne (požiadavka P4.0) preberieme od fakultného hráča a existujúcich zahraničných tímov. Hráč bude mať z každej funkcionality implementovaných viacero druhov a bude možné podľa situácie počas zápasu medzi touto funkcionalitou prepínať.

4 Hrubý návrh

V tejto kapitole je opísaný návrh rozšírenia existujúceho systému, ktorý by sme chceli v rámci tohto tímového projektu implementovať. Konkrétne sa jedná o rôzne prístupy k vylepšeniu a výberu vhodnej akcie, ktorú by mal hráč vedieť vykonať.

Ako základnú kostru pre implementáciu tohto rozšírenia sme sa rozhodli použiť hráča tímu *UTTP*. Jedná sa o fakultného hráča, pričom jeho najväčšou výhodou je jeho architektúra, ktorá disponuje vysokou úrovňou modularity a tak vnáša do kódu vysokú prehľadnosť.

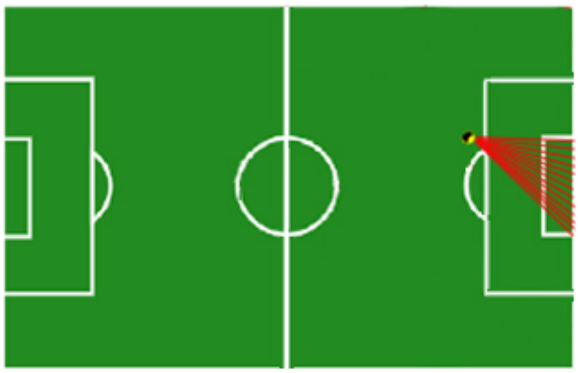
4.1 Správania, ktoré chceme prebrať

Táto kapitola je zameraná na opis konkrétnych správaní, ktoré chceme prebrať z už existujúcich tímov. V prvom kroku bude našou hlavnou snahou implementovať najjednoduchšie formy správania do nášho hráča. Po každej implementácii toto správanie overíme porovnaním so správaním, ktoré nahradilo, prípadne rozšírilo. Následne až na základe výsledku porovnávacích testov sa rozhodne, či toto správanie bude do výsledného produktu implementované, alebo nie.

V rámci tejto kapitoly sú ďalej stručne opísané jednotlivé formy správania, o ktoré chceme hráča *UTTP* rozšíriť.

4.1.1 Streľba na bránu

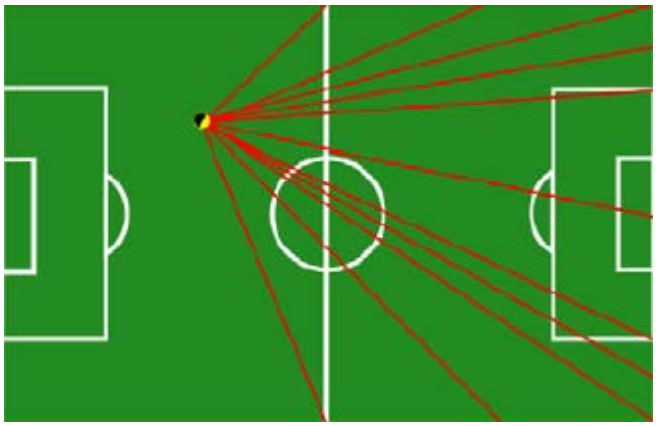
V prípade, ak by mal hráč loptu a jeho vzdialenosť od súperovej bránky by bola menšia ako 19,5 m, tak sa tento hráč rozhliadne prostredníctvom 25 bodov smerom k bránke [8]. Tieto body sú umiestnené v rovnakej vzdialenosti od seba. Postupne si tieto cieľové body prezrie od kraja k stredu (pretože je väčšia pravdepodobnosť, že brankár strelu nechytí, ak smeruje napríklad do rohu bránky). Následne ju kopne smerom, ktorým nestojí žiaden protihráč a ani jeho spoluhráč. Ošetrenie prítomnosti vlastného spoluhráča má význam z toho dôvodu, že by sa potenciálne mohol snažiť chytiť loptu smerujúcu do súperovej bránky. Takto sa zabezpečí strela na bránu s gólovým potenciálom. Celú situáciu znázorňuje obrázok 12.



Obrázok 12 Streľba na bránku [8]

4.1.2 Prihrávanie si hráčov

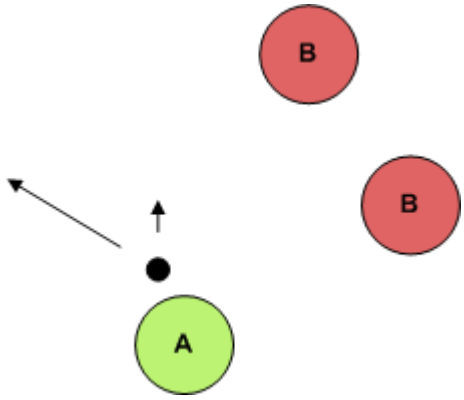
Prihrávanie by fungovalo na nasledujúcom princípe [8]. Hráč si rozdelí hraciu plochu od jeho pozície smerom k bráne na niekoľko oblastí. Toto delenie vznikne prostredníctvom priamok spájajúcich jeho aktuálnu pozíciu s určitými bodmi na ihrisku. Celú situáciu znázorňuje obrázok 13. Následne na to nájde tú časť, v ktorej je najmenej protihráčov. Ak sa v tejto časti nachádza aj jeho spoluhráč, tak tomuto spoluhráčovi prihrá loptu.



Obrázok 13 Rozdelenie hracej plochy prostredníctvom priamok [8]

4.1.3 Driblovanie

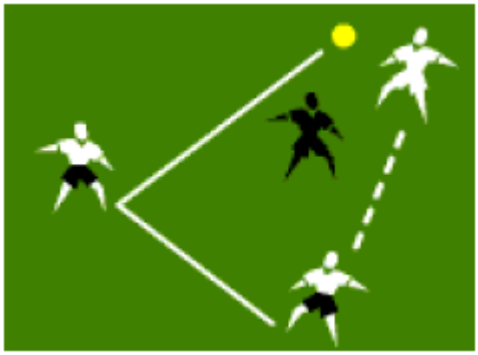
V oblasti práce s loptou použijeme driblovanie hráča implementované tímom *Mainz Rolling Brains* [3]. Okrem iného sa vyznačuje možnosťou meniť parametre driblovania počas hry, t. j. určenie relatívnej pozície lopty vzhľadom k hráčovi alebo vzdialenosť odkopu lopty od hráča atď. Hráč sa takisto snaží manipulovať s loptou v bezpečnej vzdialenosti od oponentov podľa čoho je adekvátne usmerňovaný smer pohybu s loptou a skôr spomenuté parametre driblovania. Na obrázku 14 je znázornený zjednodušený proces rozhodovania hráča o smere a vzdialenosti odkopu lopty pri driblovaní. V skutočnosti sa do úvahy berú viaceré smery a nielen dva ako je to uvedené na tomto obrázku, pričom sa vyhodnocuje vhodnosť driblovania týmito smermi.



Obrázok 14 Rozhodovanie o smere a veľkosti odkopu lopty pri driblovaní

4.1.4 Nadbiehanie hráča do voľného priestoru

Hráč, ktorý má loptu, nahrá svojmu voľnému spoluhráčovi a potom začne utekať do voľného priestoru [2]. V tomto priestore bude čakať prihrávku od spoluhráča, ktorému nahral prvý krát. Pri útočnej fáze je veľmi dôležité, aby hráč ktorý nahráva predsunutému hráčovi dával pozor na ofsajd. Táto situáciu znázorňuje obrázok 15 .



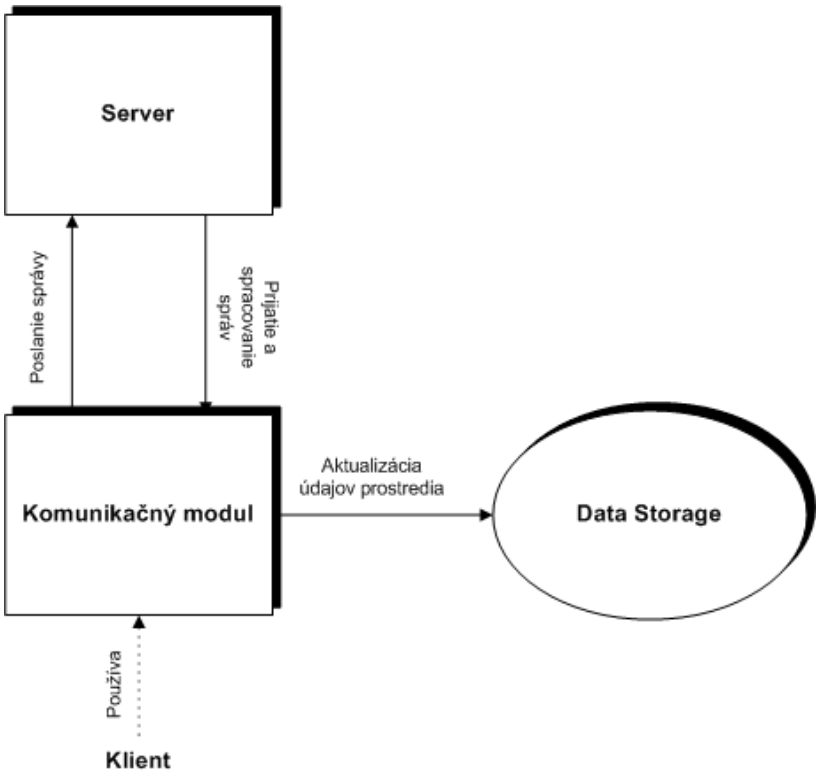
Obrázok 15 Nadbehnutie si hráča do voľného priestoru [2]

4.2 Rozšírenia existujúceho hráča o zložitejšie vlastnosti

Táto kapitola je zameraná predovšetkým na implementáciu zložitejších vlastností, ktoré by zlepšili herné správania sa nášho hráča. Šlo by napríklad o vzájomnú komunikáciu medzi hráčmi počas hry. Ďalším príkladom je implementovanie rozhodovania výberu medzi rovnakými formami správania.

4.2.1 Modul pre posielanie správ medzi hráčmi

Pre zabezpečenie komunikácie medzi hráčmi počas hry sme navrhli implementovať komunikačný modul do nášho tímu. Jeho štruktúra je znázornená na obrázku 16.



Obrázok 16 Návrh architektúry systému pre posielanie správ

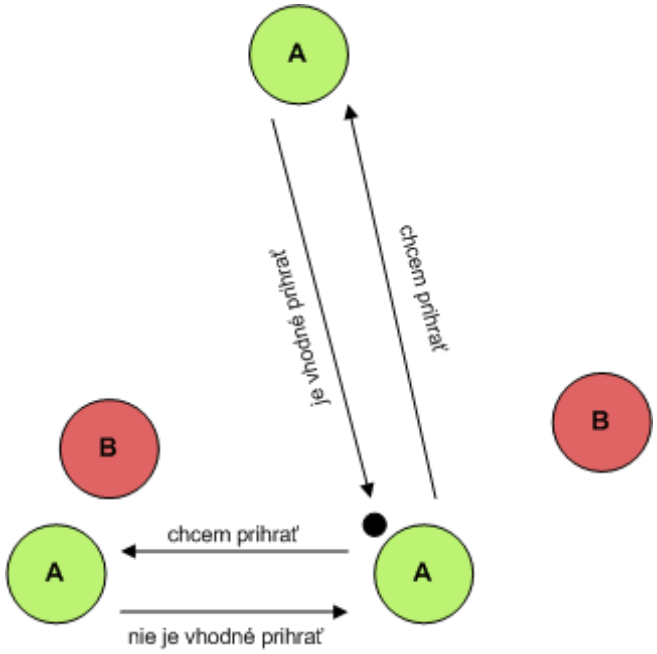
Tento modul by slúžil na posielanie a spracovávanie správ, ktoré slúžia na vzájomnú komunikáciu medzi členmi tímu. Ak má hráč záujem o informovaní svojho okolia o nejakom fakte, pošle o tom na server štruktúrovanú správu, ktorý ju následne (podľa hracích pravidiel) prepošle ostatným spoluhráčom.

Modul slúži taktiež na príjem a následné spracovanie správ od spoluhráčov. Po príchode správy sa táto správa spracuje a na základe jej obsahu sa aktualizujú príslušné údaje o prostredí v pamäti hráča (napr. pozícia lopty).

4.2.2 Posielanie správ medzi hráčmi

Posielanie správ medzi hráčmi sme sa rozhodli implementovať za účelom skvalitnenia údajov o hracom prostredí a vylepšeniu rozhodovania sa hráča počas zápasu. Hráči nachádzajúci sa blízko seba si navzájom vymieňajú údaje o prostredí alebo o priebehu hry. Prakticky to znamená, že hoci sa nejaký objekt nenachádza v zornom uhle hráča, môže ho iný hráč, ktorý pozná pozíciu tohto objektu, informovať poslaním správy obsahujúcej lokalizačné údaje o tomto objekte. Takýmto spôsobom si hráči vymieňajú nielen údaje o svojej polohe, ale i údaje o polohe lopty a pozícii protihráčov. Hráč tak má k dispozícii presnejšie údaje o svojom okolí, čo umožní jeho lepšie rozhodovanie o akcii, ktorú má vykonať v danej situácii. Druhou aplikáciou komunikácie medzi hráčmi je zlepšenie kvality hry. Hráč nachádzajúci sa v špecifickej situácii (nachádza sa v nevýhodnej situácii, v blízkosti súperovej bránky a pod.) informuje svojich spoluhráčov v jeho blízkosti o zámere prihrať loptu. Spoluhráči potom majú možnosť požadovať, resp. odmietnuť prijatie prihrávky lopty. Princíp tohto procesu je zachytený na obrázku 17. Požiadavku na prihratie lopty alebo jej odmietnutie pritom môžu hráči poslať aj sami bez predchádzajúcej deklarácie zámeru prihrať loptu.

Výmena správ medzi hráčmi je použiteľná na riešenie niektorých situácií akou je napríklad zachytenie lopty. Vzájomnou komunikáciou sa dá napríklad zabrániť tomu, aby za loptou zbytočne vyštartovalo naraz niekoľko hráčov nachádzajúcich sa v jej blízkosti. Hráč tak má opäť k dispozícii kvalitnejšie údaje o priebehu hry, ktoré môže zohľadniť pri svojom rozhodovaní.



Obrázok 17 Príklad komunikácie medzi spoluhráčmi tímu A

4.2.3 Voľba konkrétneho správania

V prípade ak hráč disponuje viacerými správaniami toho istého druhu, je nutné aby vedel vybrať správanie, ktoré je pre aktuálnu situáciu najvhodnejšie.

Tento výber je možné realizovať napríklad na základe nasledujúcich pravidiel:

- V prípade ak daná forma správania bude rapídne uberať na svojej účinnosti, prestane sa toto správanie používať
- Každé správanie bude mať definované obmedzenia, kedy je vhodné ho použiť. Napríklad použi typ driblovania číslo 1 ak sa v tvojej blízkosti nenachádza žiaden protihráč, inak použi driblovanie číslo 2

Bude teda existovať hodnotiaca funkcia, ktorá ohodnotí jednotlivé rovnaké správania. Každé jedno z týchto rovnakých správaní bude mať definované hranice, kedy je ktoré vhodnejšie. Nakoniec sa aplikuje jedno z nich na základe ohodnotenia.

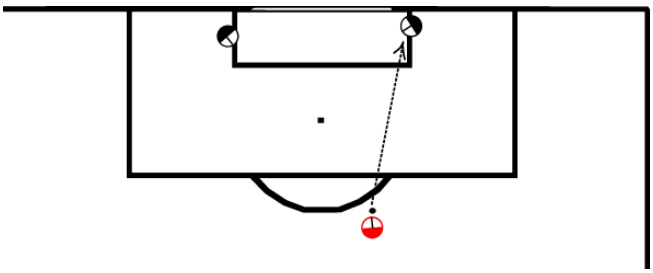
5 Prototyp

V tejto kapitole sa venujeme popisu vytvoreného prototypu, ktorý bol vytvorený na základe hrubého návrhu. Cieľom prototypu je zlepšiť základné správanie sa hráčov, napríklad streľba na bránu, prihrávanie si hráčov. V prototype sme upravili správanie sa obrancov tak, aby sa im zbytočne neničila stamina, keď sú v obrannom pásme a lopta je na súperovej polovičke ihriska. Taktiež sme sa zamerali na vylepšenie streľby hráča na bránku vo vzdialenosti menšej ako 22 metrov od bránky. Podarilo sa nám implementovať základnú kostru komunikácie medzi hráčmi aj vďaka využitiu existujúcich štruktúr.

5.1 Streľba na bránu

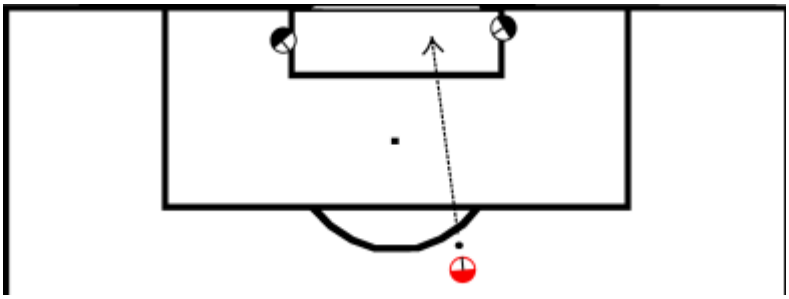
Snažili sme sa vylepšiť streľbu na súperovu bránu implementovaním správania GoalKick, ktoré sme chceli prebrať z hráča *Dirty Dozen*. Po analýze existujúcich zdrojových kódov sme našli správanie podobné správaniu GoalKick, ktoré sme chceli implementovať. Toto správanie však nebolo hráčom UTTP využívané. My sme sa rozhodli dané správanie použiť v našom hráčovi.

Hlavnou myšlienkou pôvodného správania bolo kopanie lopty čo najbližšie k tyčke bránky v závislosti od pozície súperovho brankára. Toto správanie ale vôbec nezohľadňovalo pozíciu ostatných hráčov pred bránkou. V prípade ak by oba tyčky boli obsadené nepriateľom, strela by stále sledovala k jednej z tyčiek a nie smerom k strede bránky – tak, ako by to bolo najvhodnejšie. Táto situácia je zobrazená na obrázku 18.



Obrázok 18 Streľba na bránku smerom k tyčke nezohľadňujúc pozíciu nepriateľa.

My sme toto správanie zlepšili implementovaním zohľadnenia nepriateľa pred bránkou. V hore uvedenom prípade náš hráč strelí loptu do stredu bránky, namiesto streľby k jednej z tyčiek. Túto situáciu znázorňuje obrázok 19.



Obrázok 19 Streľba na bránku zohľadňujúc pozíciu nepriateľa.

Ďalej sme dané správanie upravili zvýšením hodnoty vzdialenosti, z ktorej môže hráč kopať na súperovu bránku. Túto hodnotu sme zvýšili o 5 metrov z hodnoty 17 na hodnotu 22 metrov. Pri analýze hry sa hráč totiž snažil dostať až príliš blízko k súperovej bránke, pričom bol už vo vhodnej pozícii na strelu.

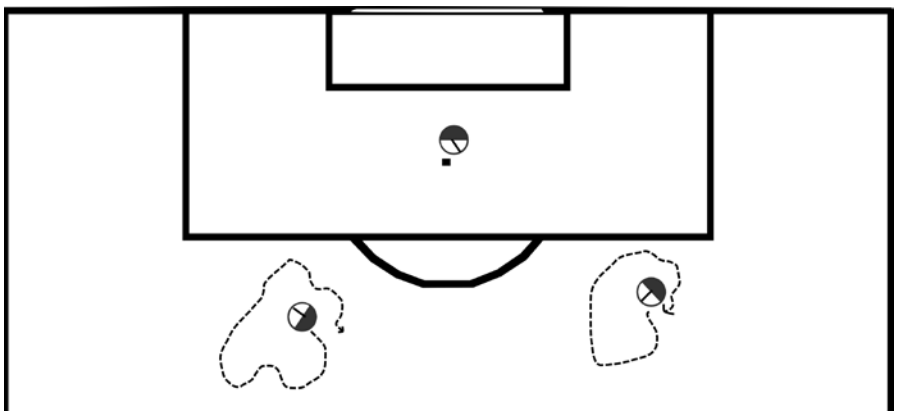
5.2 Drobné opravy kódu ako nástroj pre vylepšenie hráča

Pri prototypovaní sme zistili, že hráč dokáže pri drobných úpravách pôvodného zdrojového kódu podávať oveľa lepšie výsledky, ako predtým. Takéto úpravy sú jednoduché na implementáciu, avšak prípadné zmeny treba vypozorovať z priebehu hry a následne nájsť v pôvodnom zdrojovom kóde. Čo nemusí byť triviálne. Avšak chceme sa pri implementácii vydať aj touto cestou, lebo pôvodná funkcionalita hráča tímu UTTP je často dobre naimplementovaná, len ju treba odladiť.

5.2.1 Vylepšenie výdrže obrancov

Pri analýze zápasov tímu UTTP sme si všimli nelogické správanie sa obrancov. Pri väčšej vzdialenosti lopty od bránkoviska obrancovia začnú vykonávať nezmyselné pohyby. Hráči

neustále vybiehajú smerom od bránkoviska a zasa sa k nemu vracajú, čo im míňa veľkú časť energie. Situáciu znázorňuje obrázok 20.



Obrázok 20 Nelogické správanie sa obrancov.

Chyba v správaní obrancu bola spôsobená nedokonalosťou pôvodného zdrojového kódu. Hráči pri istej vzdialenosti lopty od bránkoviska sa mali vrátiť iba na určenú pozíciu. Avšak túto pozíciu nevedeli presne trafiť.

Táto chyba bola nájdená v metóde:

```
void RunToPositionBehaviour::Behave(Point position, int speedMode).
```

A to konkrétne v podmienke, ktorá zisťovala, či hráč ešte nie je na určenom mieste:

```
if(diff.GetLength() > 0) ...
```

Táto podmienka bola neustále pravdivá, pretože vzdialenosť vždy bola väčšia ako 0, vzhľadom na presnosť, s ktorou sa hráč môže pohybovať po ihrisku. Upravili sme podmienku:

```
if(diff.GetLength() > 1) ...
```

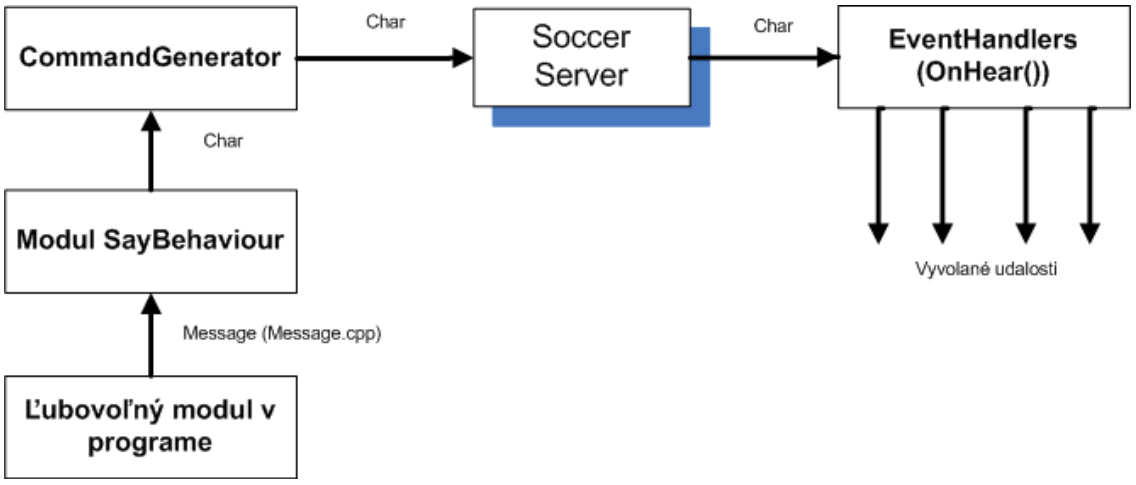
Vzdialenosť 1 meter od požadovanej pozície považujeme za dostatočnú vzhľadom na veľkosť ihriska. Po úprave podmienky obrancovia prestali zbytočne míňať energiu.

5.3 Komunikácia hráčov

Komunikácia medzi hráčmi je realizovaná pomocou krátkych správ, ktoré si medzi sebou navzájom vymieňajú. Štruktúra správy nie je pevne daná, jediným obmedzením je nutnosť použiť na jej zakódovanie danú množinu znakov. Formát správy je nasledovný:

[3 B – označenie správy] [1 B – typ správy] [1 B – identifikácia prijímateľa] { [1 B - obsah][hodnota obsahu] } x N

V ľubovoľnom module máme možnosť použiť verbálnu komunikáciu a to pomocou inštancie modulu SayBehaviour. Správu, ktorú chceme poslať, musíme však najskôr vytvoriť. Na to slúži trieda Message, ktorá je určená na jednoduchú manipuláciu s obsahom prenášaných správ. Pomocou metódy getRawMessage(), implementovanej v triede Message, skonvertujeme správu do podoby, ktorú je možné použiť na zavolanie metódy realizujúcej vyvolanie príkazu Say na soccer serveri. Soccer server následne preposiela prijaté správy hráčom, ktorý majú v triede EventHandlers (metóda OnHear()) implementovanú spracovanie týchto správ a rozhodovaciu logiku na základe obsahu prijatých správ. Štruktúra posielania správ je znázornená na obrázku 21.



Obrázok 21 Štruktúra posielania správ medzi hráčmi.

Uvedená komunikácia je použitá na vzájomné informovanie sa hráčov o situácií na ihrisku a na riešenie niektorých situácií, predovšetkým skvalitnenie nahrávok medzi nimi.

V prototype sme implementovali základnú kostru takejto komunikácie, pričom sme na overenie funkčnosti museli použiť vlastnú štruktúru správ. Tá je zložená z údaju o type správy (požiadavka, odpoveď) a niekoľkých premenných, ktorým sú priradené hodnoty. Zoznam premenných, ich označenie a význam je vopred zadefinovaný v zdrojových kódoch a preto vieme jednoznačne extrahovať tieto hodnoty z prijatej správy.

Predpokladáme, že v budúcnosti sa štruktúra a obsah posielaných správ zmení a to v súvislosti s dôkladným testovaním vplyvu a významu vymieňaných údajov na priebeh hry. Základná kostra implementácie spočíva vo vyslaní správy spoluhráčom nachádzajúcim sa v blízkosti hráča, ktorý má pod kontrolou loptu. Na základe prijatej správy môžu títo hráči spätne poslať požadovanú informáciu, napr. výhodnosť postavenia pred bránkou na strelenie gólu alebo vhodnosť realizovania prihrávky ich smerom. Tieto vyhodnocovacie funkcie však budú predmetom našej práce až v letnom semestri a preto sme im pri vytváraní prototypu nevenovali veľkú pozornosť. Našim zámerom bolo predovšetkým vytvoriť základ, na ktorom budeme stavať ďalšiu prácu.

5.4 Testovanie

Testovanie prototypu prebiehalo v závislosti od typu správania, ktoré sme implementovali. Na každý z týchto typov správania sme použili inú testovaciu techniku pre získanie čo najpresnejšieho porovnanie nášho hráča Kukuričné deti s pôvodným hráčom tímu UTTP.

5.4.1 Testovanie streľby na bránu

Správnosť streľby na bránku sme testovali špeciálne len na strelecké situácie. Najskôr sme testovali vzdialenosť, z ktorej daný hráč strieľa na bránku súpera. Test prebehol v klasickom soccer monitore, pričom sme vytvorili situáciu, keď má nepriateľ vo svojom tíme iba brankára. My sme na svojej strane mali iba brankára a útočníka. Vizuálne sme sledovali jeho strely na bránku, pričom bolo jasne vidieť, že vzdialenosť od súperovej bránky, z ktorej náš hráč strieľal, sa zvýšila.

Pre otestovanie streľby lopty pomedzi hráčov (zohľadnenie ich pozície) sme použili trénera pre vytváranie situácií, ktoré môžu nastať. Jednalo sa o napodobnenie situácie opísanej vyššie v tomto dokumente. Z testu vyplynulo, že nami implementované správanie skutočne berie pri streľbe na bránu do úvahy pozíciu súperových hráčov.

5.4.2 Testovanie vylepšenie výdrže obrancov

Testovanie regenerácie staminy u hráčov v obrane prebiehalo počas zápasu tímu UTTP a Kukuričné deti. Počas priebehu stretnutia týchto dvoch tímov sa sledovala hodnota staminy obrancov na jednej a druhej strane súčasne. Sledovanie staminy prebiehalo v čase, keď sa lopta zdržiavala dlhší čas v strede hracej plochy. Takto mala totiž jedna aj druhá strana možnosť dobíjať si svoju staminu v obrane. Odpozorované výsledky je možné vidieť v tabuľke 1.

Tabuľka 1 Meranie staminy u obrancov.

	Počet výskytov prípadov v hre			
	Tím UTTP		**Tím Kukuričné deti**	
	1 unavený obranca	**2 unavení obrancovia**	**1 unavený obranca**	**2 unavení obrancovia**
1.polčas	**5**	**1**	**3**	**0**
2.polčas	**3**	**2**	**2**	**0**

Z uvedených výsledkov je vidieť, že nami implementovaná zmena bola účinná a pomohla tak k zvýšeniu výkonu obrany nášho tímu.

5.4.3 Testovanie komunikácie hráčov

Testovanie komunikácie medzi hráčmi na ihrisku sme realizovali prostredníctvom výpisov komunikácie medzi hráčmi počas zápasu v okne konzoly. Sledovali sme text správy, akému hráčovi bola určená a ako mal na ňu reagovať.

Napríklad sa jednalo o text správy od hráča s číslom 2 na hráča s číslom 3. V kontrolnom výpise sa objavila táto správa:

1. U hráča č. 3 sme pozorovali tento výpis: „Prišla správa od hráča č. 2“

2. U hráča č. 2 sme pozorovali tento výpis: „Prišla odpoveď od hráča č. 3“

Uvedenú komunikáciu sme samozrejme testovali viackrát a vždy sme mohli badať, že hráč ovládajúci loptu vysiela do svojho okolia zvukové správy, ktoré jeho spoluhráči zachytávajú a reagujú na ne poslaním odpovede. Tá je určená pre iniciátora komunikácie a v budúcnosti bude obsahovať požadované údaje.

5.5 Zhodnotenie prototypu

Prácou na prototype sme sa podrobnejšie všetci oboznámili so štruktúrou kódu nášho hráča a úspešne sme implementovali, prípadne pozmenili, niekoľko štruktúr. Pri implementácii sme sa stretli s viacerými problémami, napr. s chybami pri linkovaní súborov, s nepoužívanými metódami, či dokonca s nezmyselným správaním.

Samozrejme nie všetky správania, ktoré by bolo potrebné odstrániť, pozmeniť, prípadne doplniť sme boli schopní v tomto krátkom čase opraviť. Ako veľmi nevyhnutné bude potrebné zlepšiť, prípadne úplne vymeniť správanie prihrávania si lopty medzi hráčmi a správanie sa obrancov.

Celkovo, aj na základe vyššie uvedených zistení, sme zhodnotili, že na hráčovi nášho tímu - Kukuričné deti, vychádzajúceho z tímu UTTP, bude potrebné vykonať viac úprav existujúceho kódu, ako sa predpokladalo.

6 Opis riešenia

V tejto kapitole sa venujeme opisu vlastností, ktoré sme u pôvodného hráča UTTP vylepšili v našom hráčovi Kukuričné deti. Ďalej uvádzame použité implementačné prostredie a zmeny oproti návrhu.

6.1 Výber implementačného prostredia a jazyka

Výber implementačného jazyka bol v našom prípade jednoduchý, keďže sme vychádzali z už existujúcich zdrojových súborov hráča UTTP. Tento hráč je implementovaný v jazyku C++. V pôvodnej dokumentácii sa uvádzalo, že tento hráč je aj multiplatformový. Je možné ho skompilovať a spustiť pod operačným systémom *Windows*, ako aj pod operačným systémom *Linux*. Pri snahe rozbehať tohto hráča pod systémom Linux sme narazili na problémy s *Makefile* súborom. Problém sme neboli schopní odstrániť.

Ako implementačné prostredie sme použili *Visual Studio* 2008 (VS). Pre toto vývojové prostredie sme sa rozhodli z dôvodu predchádzajúcich skúseností a faktu, že sa jedná o najnovšie vývojové prostredie pre prácu s C++ na platforme Windows. Pôvodného hráča sme preto museli prekonvertovať z prostredia *VS 2005* na prostredia *VS2008*.

6.2 Zmeny oproti návrhu

Od pôvodného návrhu, hlavne spôsobu vylepšovania hráča preberaním správaní, sme sa značne odchýlili. Na pôvodnom hráčovi UTTP sme objavili množstvo chybového správania. Oprava tohto správania sa pre nás stala prvoradou a všetky ďalšie navrhované zmeny v časti návrhu sme odložili.

Oprava hráča, ktorú sme realizovali, sa celkovo týkala jeho prihrávania, driblovania, obzerania sa po ihrisku. Ďalšie opravy sa týkali vykopávania lopty brankárom a rozohrávania autových situácií.

Tieto opravy je možné vykonať aj prebratím správaní z iných tímov, tak ako sme to navrhovali v sekcii návrh. Tento postup by bol ale oveľa zložitejší vzhľadom na základné možnosti, ktoré ponúkala základná funkcionalita hráča UTTP.

7 Realizácia riešenia

Táto kapitola opisuje implementované zmeny a rozšírenia hráča Kukuričné deti. Účelom bolo zlepšiť schopnosti hráča a zvýšiť jeho hernú aktivitu.

7.1 Zmena vo vykopávaní lopty brankárom

V pôvodnej verzii hráča bolo správanie brankára pri vykopávaní lopty značne zjednodušené. Po zachytení lopty sa brankár premiestnil na pevne stanovenú pozíciu, z ktorej vykopol loptu na vopred zadefinovaný bod. Nezohľadňoval aktuálnu situáciu na ihrisku, t. j. rozmiestnenie hráčov pri vykopávaní lopty.

Miesto výkopu lopty sa pôvodne určovalo tak, že sa vybral jeden z dvoch preddefinovaných bodov nachádzajúcich sa v rovnakej vzdialenosti nad a pod x-ovou osou ihriska v priestore bránkoviska. Vybral sa pritom najbližší z týchto pevne stanovených bodov k miestu zachytenia lopty brankárom. K miestu výkopu lopty boli takisto vopred určené pozície v priestore ihriska, kam výkop lopty smeroval. Tieto pozície boli určené tak, aby výkop lopty smeroval čo najďalej do stredu ihriska popri hornom, resp. dolnom okraji ihriska.

Jadro správania brankára pri vykopávaní lopty sa nachádza v module `KickOffBehaviour`, konkrétne v súbore `KickOffBehaviour.cpp`. V tomto súbore sme takisto realizovali tieto zmeny správania:

- **určenie miesta výkopu lopty,**
- **určenie rýchlosti výkopu lopty.**

Pod určením miesta výkopu lopty máme na mysli nielen pozíciu v bránkovisku, z ktorej brankár loptu vykopáva ale i destináciu, t. j. kam lopta smeruje. Na určenie najvhodnejšieho miesta výkopu lopty sa pre každú kombináciu bodov, zloženú z miesta výkopu lopty (niekoľko bodov nachádzajúcich sa na pevne zvolenej vertikálnej úsečke v priestore bránkoviska) a miesta, kam sa má lopta prihrať (pozície spoluhráčov), určí váha. Tá vyjadruje vhodnosť výkopu lopty týmto

smerom. Na výpočet tejto váhy sme do modulu `KickOffBehaviour` doplnili funkciu `evaluateKickOff`. Platí pravidlo, že čím vyššia je táto váha, tým je vhodnejšie týmto smerom loptu vykopnúť. Na realizovanie výkopu sa preto vyberie dvojica bodov s najvyššou váhou. Pri výpočte váhy sa berie do úvahy viacero faktorov: uhlová vzdialenosť k najbližšiemu nepriateľovi od miesta výkopu (čím vyššia, tým vyššia je váha), počet nepriateľov v okolí miesta kam lopta smeruje (čím menej, tým vyššia je váha), vzdialenosť k miestu výkopu lopty (čím ďalej, tým vyššia je váha).

Treba pripomenúť, že z rozhodovania brankára pri výkope lopty sú vylúčení hráči, ktorí sa nachádzajú v priestore medzi brankárom a jeho bránou a hráči, v ktorých blízkosti sa nachádza nepriateľ. Ak nie je možné prihrať ani jednému spoluhráčovi, tak sa aplikujú pôvodné funkcie na vykopávanie lopty, ktoré sa však v pôvodnom kóde používali len výnimočne. Konkrétne ide výkop lopty realizovaný použitím funkcií využívajúcich gradientové mapy. Brankár v tomto prípade loptu kopne do priestoru tak, aby lopta nesmerovala na nepriateľa a šla čo najďalej do priestoru ihriska.

Pri výkope lopty na konkrétneho hráča sa zohľadňuje aj jeho vzdialenosť od brankára. *Rýchlosť (sila) výkopu lopty* je stanovená na sto percent pri výkope za vopred určenú medznú hranicu (je to vlastne pevne daná vzdialenosť od brankára v metroch). Ak sa ale hráč nachádza príliš blízko, tak lineárnou interpoláciou sa rýchlosť zníži. Vďaka tomu sa zvýšia šance hráča na spracovanie lopty.

7.2 Hľadanie lopty hráčom

Počas hry sa bežne stávalo, že hráč bez lopty nemal aktuálne údaje o lopte počas väčšieho počtu cyklov za sebou (niekedy aj viac ako 20). Ak by aktuálne údaje o lopte nemal viac 12 cyklov, tak preskúma svoje okolie za účelom jej nájdenia. Táto časť kódu sa nachádza v triede `WithoutBehaviour`. Využili sme pritom funkcionalitu, ktorá už bola implementovaná v pôvodnom hráčovi v module `SearchForBallBehaviour`.

V triede `WithoutBehaviour` sme navyše implementovali aj preskúmanie okolia hráča rotáciou okolo jeho osi, ktoré sa aplikuje pri každom prechode do hracieho módu iného ako `Play_On` (napr. vykopávanie lopty, zahrávanie autu atď.). Vďaka tomu získa hráč čerstvé informácie o svojom najbližšom okolí.

7.3 Rozšírenie modelu sveta

Umiestnenie rozšírenia v projekte:

./Player/Player/World/Enemy.cpp
./Player/Player/World/Enemy.h
V modeli sveta *World* je implementovaná trieda *Enemy*, ktorá zapuzdruje nové vnímanie sveta hráčom. V tejto triede je ďalej vytvorená podtrieda *Triangle* reprezentujúca pohľad hráča.

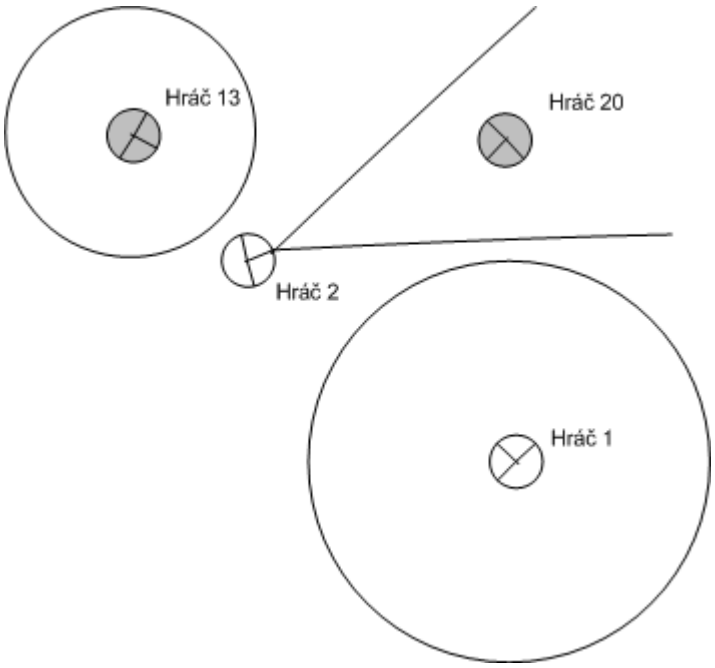
Každý hráč v hre sa pozerá po ihrisku, pričom každé jedno takéto pozretie predstavuje pohľad hráča. Pohľad je v podstate trojuholník, ktorý môže byť dvoch rôznych veľkostí (pohľad pod uhlom 45 a 90 stupňov), alebo tvaru polkruhu (pohľad pod uhlom 180 stupňov). Množinu pohľadov zapuzdruje trieda *Triangle*, ktorá je podtriedou triedy *Enemy*.

Ďalšie zmeny, ktoré si implementácia rozšíreného modelu sveta vyžiadala, sa týkali doplnení volaní do triedy *World*. Jednalo sa o vytvorenie inštancie triedy *Enemy* a o aktualizáciu jej dátových štruktúr. Táto trieda využíva aj existujúce štruktúry, ktoré poskytujú informácie o stave sveta, a to:

- *World->enemies* – Pole reprezentujúce zoznam protihráčov na ihrisku s informáciou o ich polohe, atď.
- *World->friends* – Pole reprezentujúce zoznam spoluhráčov na ihrisku s informáciou o ich polohe, atď.

Trieda Enemy

Hlavnou úlohou triedy Enemy je vnímanie sveta hráčmi na vyššej úrovni. Hráč vníma nepriateľov a spoluhráčov prostredníctvom kružníc, so stredom v mieste ich posledného videnia (miesto, v ktorom náš hráč naposledy videl iného hráča). Polomer kružnice reprezentuje vzdialenosť, kam mohol odpozorovaný hráč najďalej dobehnúť po tom, ako sme ho prestali pozorovať. Príklad takéhoto vnímania sveta ilustruje obrázok 22.



Obrázok 22 Príklad stavu sveta v nadstavbovom modeli.

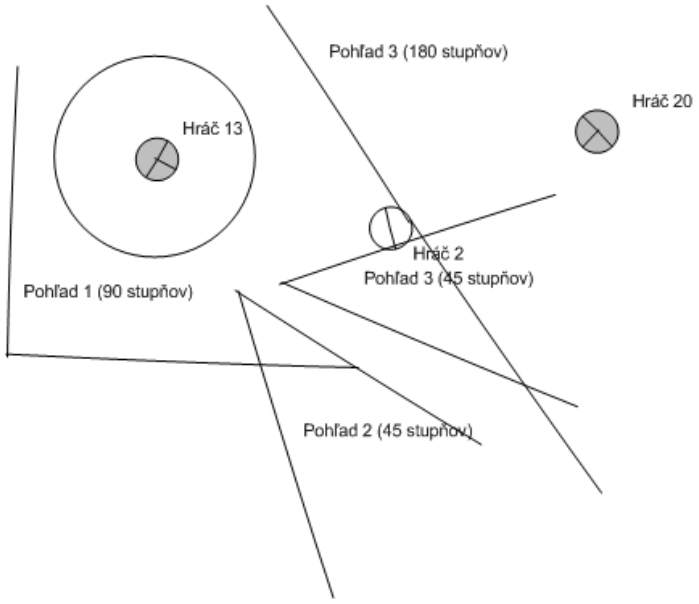
Na tomto príklade je vidieť hráča s číslom 2, ktorý vo svojom pohľade (v podstate trojuholník) vidí protihráča s číslom 20. Hráč číslo 2 má ďalej informáciu o protihráčovi s číslom 13 a o svojom spoluhráčovi s číslom 1. Informácia o hráčoch s číslami 13 a 1 je stará niekoľko cyklov, z tohto dôvodu sú okolo týchto hráčov kruhy ich ďalšieho možného pohybu za určitý počet cyklov dozadu. Podľa veľkosti kruhu je vidieť, že informácia o našom spoluhráčovi (hráč s číslom 1) je podstatne staršia, ako informácia o protihráčovi (hráč s číslom 13). Kruh ale nie je

možné zväčšovať do nekonečna. Po uplynutí 15 cyklov, od posledného videnia hráča, sa informácia o zaznamenanom pozorovaní stráca a kruh zanikne.

Hlavným dôvodom takéhoto vnímania je zlepšiť prihrávanie si hráčov medzi sebou, ktoré pôvodne bolo na veľmi slabej úrovni. S týmto problémom sa spájali hlavne nepresné prihrávky, prihrávky nepriateľovi a slabá informovanosť hráča o jeho spoluhráčoch.

Trieda *Triangle*

Trieda *Enemy* obsahuje podtriedu *Triangle*, ktorá reprezentuje pohľad hráča. Dynamický zoznam typu tejto triedy predstavuje množinu pohľadov hráča, ktoré boli počas hry zaznamenané. Podobne ako pri kruhoch, aj pri týchto pohľadoch sa určuje ich vek. Pohľady staršie ako 15 cyklov sú automatický z množiny pohľadov zmazané. Príklad štyroch pohľadov hráča na ihrisku zobrazuje obrázok 23.



Obrázok 23 Rôzne pohľady hráča v hre**.**

Na tomto príklade je vidieť hráča s číslom 2 a jeho štyri rôzne pohľady s rôznymi veľkosťami uhlov. Pohľadom číslo 1 bol zbadaný hráč s číslo 13. Ďalej nasleduje niekoľko ďalších pohľadov

toho istého hráča. Celkovo bolo hráč 2 vykonal 4 pohľady rôznej veľkosti (90 stupňov, 2x 45 stupňov a raz 180 stupňov).

Opis hlavných metód triedy Enemy

Pomocou metód triedy *Enemy* je modifikovaná aj podtrieda Triangle, konkrétne niektoré jej verejné premenné. Ďalej je uvedený opis funkcionality najzákladnejších metód tejto triedy. Pre podrobnejší opis je nutné pozrieť komentáre v zdrojových kódoch.

```
void removeInvalidTriangles();
void addTriangle();
```

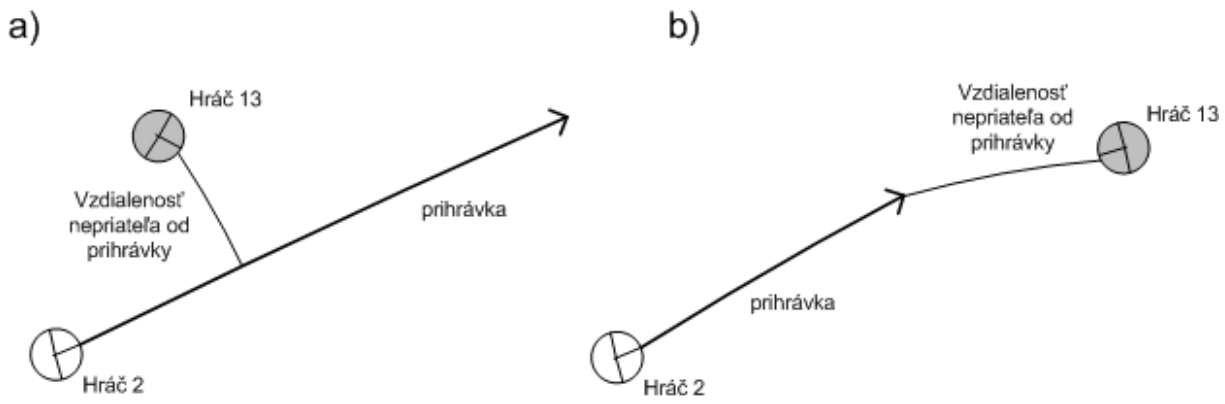
Tieto dve metódy pridávajú a odoberajú pohľady z množiny pohľadov *triangle* (označenie *triangle* predstavuje názov dynamického poľa v triede *Enemy*, ktoré je typu *Triangle*) Jedná sa o dynamické pole maximálnej veľkosti 100. Znamená to, že maximálny počet pohľadov jedného hráča za čas 15 cyklov je 100. Toto číslo je dostačujúce.

```
bool canPassSafely(Point& target, float probability = 0.75f);
```

Táto metóda vracia pravdu, ak je možné prihrať loptu z aktuálnej pozície hráča do bodu *target* s pravdepodobnosťou nad danou hranicou *probability*. Po zavolaní tejto metódy je zavolaná metóda `getPassProbabilitySuccess(…).`

```
float getPassProbabilitySuccess(Point& origin, Point& target);
```

Táto metóda vracia pravdepodobnosť úspešnosti prihrávky z pozície *origin* na pozíciu *target*, zadaných ako parametre tejto metódy. Hlavnou časťou je výpočet vzdialenosti úsečky prihrávky od pozície nepriateľa v čase *t*. Tento výpočet je potrebné zohľadniť v dvoch rôznych situáciách, ktoré zobrazuje obrázok 24.



Obrázok 24 Dve možnosti vzájomnej pozície prihrávky a protihráča. Prihrávku predstavuje orientovaná šípka, pozícia nepriateľa je stred hráča s číslom 13.

Situácia *a)* a *b)* ilustrujú dva rôzne prípady výpočtu vzdialenosti nepriateľa od našej prihrávky. V prípade *a)* je nutné získať vzdialenosť Pytagorovou vetou, v prípade *b)* sa jedná o vzdialenosť dvoch bodov (*target* a pozícia hráča s číslom 13).

```
Point getPassTargetPosition();
```

Metóda vráti pozíciu, na ktorú je najvhodnejšie prihrať z aktuálnej pozície hráča. V prípade ak takáto pozícia neexistuje, je vrátená pozícia (-500, 500). Nájdenie bodu zohľadňuje nasledujúce:

- Pozícia videných nepriateľov
- Pozícia videných priateľov
- Čas posledného videnia priateľa
- Čas posledného videnia nepriateľa
- Vzdialenosť priateľa od seba
- Či sa jedná o prihrávku smerom vzad

Celkovo sa najúspešnejšia nájdená pozícia určí na základe jej ohodnotenia. Ohodnotenie sa vypočítavá penalizáciou alebo odmenou na základe vyššie vymenovaných kritérií. Konkrétny spôsob vyhodnocovania je opísaný v zdrojovom kóde tejto metódy.

```
Point getPassTargetPositionWhenDefending();
```

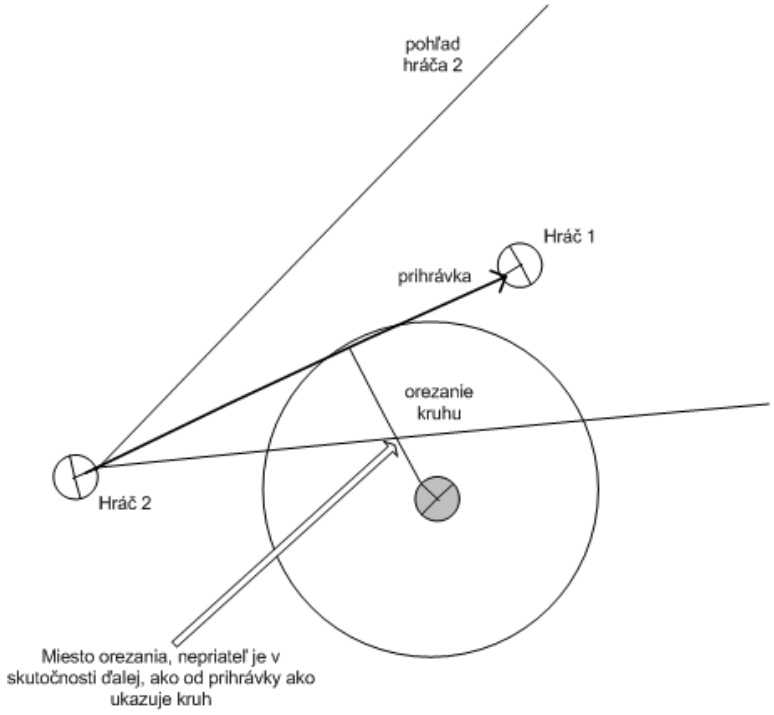
Podobne ako metóda `getPassTargetPosition()`, ale pri výbere najvhodnejšieho bodu miesta prihrávky zohľadňuje fakt, že lopta je v našej obrannej polovici a súper je nasťahovaný do nášho územia.

```
Point getPassTargetPositionForQuickOffence();
```

Podobne ako metóda `getPassTargetPosition()`, ale pri výbere najvhodnejšieho bodu miesta prihrávky zohľadňuje, či nie je možné niektorého hráča vystrčiť von, smerom do útoku z našej obrannej polovice.

```
float recalculateEnemyCircleBorders(const Myself* me, const Player* enemy, const Point* target);
```

Vždy ak pohľad nášho hráča (v podstate trojuholník alebo polkruh) pretne kružnicu, získame novú informáciu o pozícii nepriateľa na ihrisku. Príkladom je, ak v cykle 1000 sme videli nepriateľa na pozícii P(x, y). V cykle 1007 bude táto informácia stará 7 cyklov a preto bude okolo pozície P kruh s príslušným polomerom. V cykle 1005 náš hráč ale vykonal pohľad a zistil prienik kruhu s nepriateľom. V tomto prieniku ale protihráč, ktorého sme videli v cykle 1000, nebol. Z tohto dôvodu je možné polomer kruhu skrátiť o príslušnú vzdialenosť. Táto informácia nám môže pomôcť pri prihrávaní lopty spoluhráčovi. Príklad takéhoto orezania je vidieť na obrázku 25.



Obrázok 25 Príklad orezania kruhu pohľadom**.** Miesto orezania vzdialenosti nepriateľa od prihrávky ukazuje dvojitá šípka. Prihrávku predstavuje orientovaná šípka od hráča s číslom 2 smerom k hráčovi s číslom 1.

Na tomto zjednodušenom príklade je vidieť, ako orezanie pomohlo k prihrávke lopty spoluhráčovi. V tomto prípade sme orezali vzdialenosť nepriateľa od miesta prihrávky, čo umožnilo, aby metóda `canPassSafely(...)` vyhodnotila prihrávku za úspešnú.

7.4 Správanie sa hráča s loptou

Ide o úpravu správania sa hráča v prípade, ak má pod kontrolou loptu. Úprava sa týkala zjednodušenia rozhodovacieho stromu a odstránenie niektorých zúfalých akcií. Zúfalá akcia znamená, že hráč loptu odkopne náhodne, pretože nevie s ňou nič iné spraviť (žiadna iná akcia v rámci stromu nemá splnené podmienky). Ďalšia úprava sa týkala doplnenia rozhodovania sa

hráča na základe vytvorenej nadstavby modelu sveta. Bol integrované obranné správania sa hráča, rýchle útočné správanie a prihrávanie. Viacero starých správaní bolo odstránených.

Dôvody pre zmenu v rozhodovacom strome:

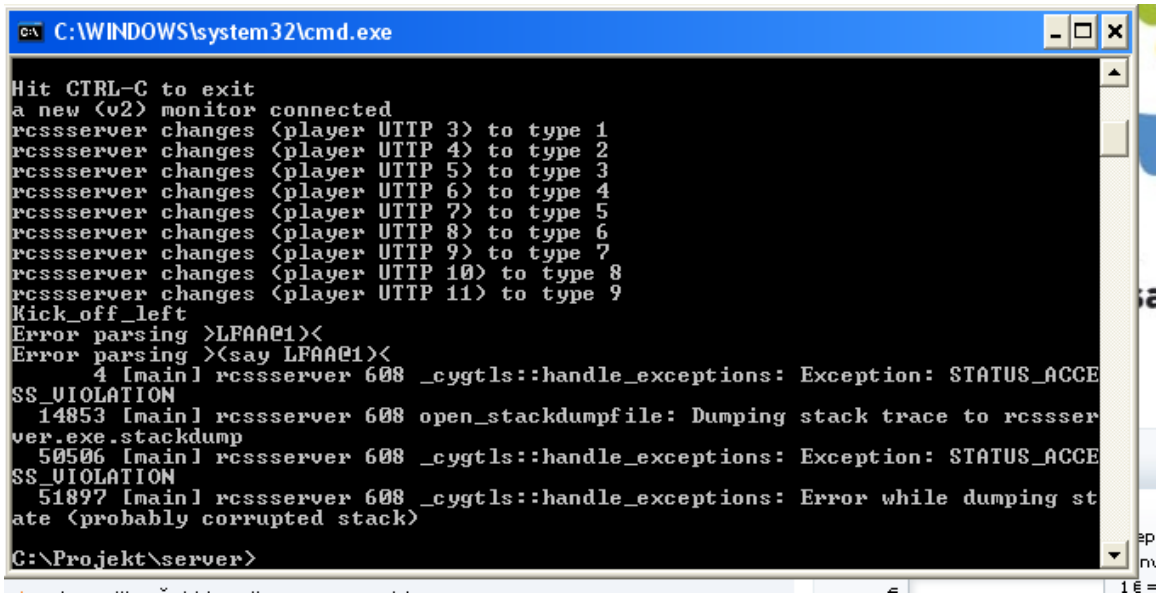
- Snaha o zjednodušenie rozhodovacieho stromu
- Odstránenie nadbytočných a nezmyselných podmienok a akcií
- Integrácia vyššie opísaných vylepšení hráča

7.5 Prechod na server 13.0.2

RoboCup finále sa uskutoční tento rok v Grazi, bude bežať na novom serveri (13.0.X). Tento nový server prináša mnohé vylepšenia okrem iného aj vyššiu rýchlosť simulácie. Preto sa aj fakultná súťaž RoboCup uskutoční na tejto novej verzii serveru. Prebraný hráč UTTP, bol ale skúšaný a robený pre nižšiu verziu servera. Preto bolo nevyhnutné otestovať hráča pod týmto novým serverom. Server je robený v dvoch verziách jedna pre Windows a druhá pre Linux. Testovanie prebiehalo nasledovným spôsobom, najprv bol simulovaný zápas medzi hráčom UTTP a tímu jahodových princov. Táto simulácia sa ani nezačala, lebo hráč jahodových princov hneď padol. Hráč UTTP nepadol. Preto bolo treba získať hráča, čo by bežal na novom serveri bez pádu a túto podmienku splnil hráč Loptošov.

Po spustení simulácie hry medzi hráčom UTTP a Loptošmi, hráči hrali uspokojivo a všetko vyzeralo v poriadku, no stala sa prekvapivá vec. Monitor sa zastavil a padli všetci hráči aj so serverom zhruba v 800tom cykle.

Pád servera sa prejavuje aj keď hrá tím UTTP osamote bez protihráčov, ďalej bola chyba testovaná už len týmto spôsobom. Pádu serveru predchádzali rôzne situácie, sú priložené dva obrázky.



Obrázok 26 Pád serveru 13.0.2.

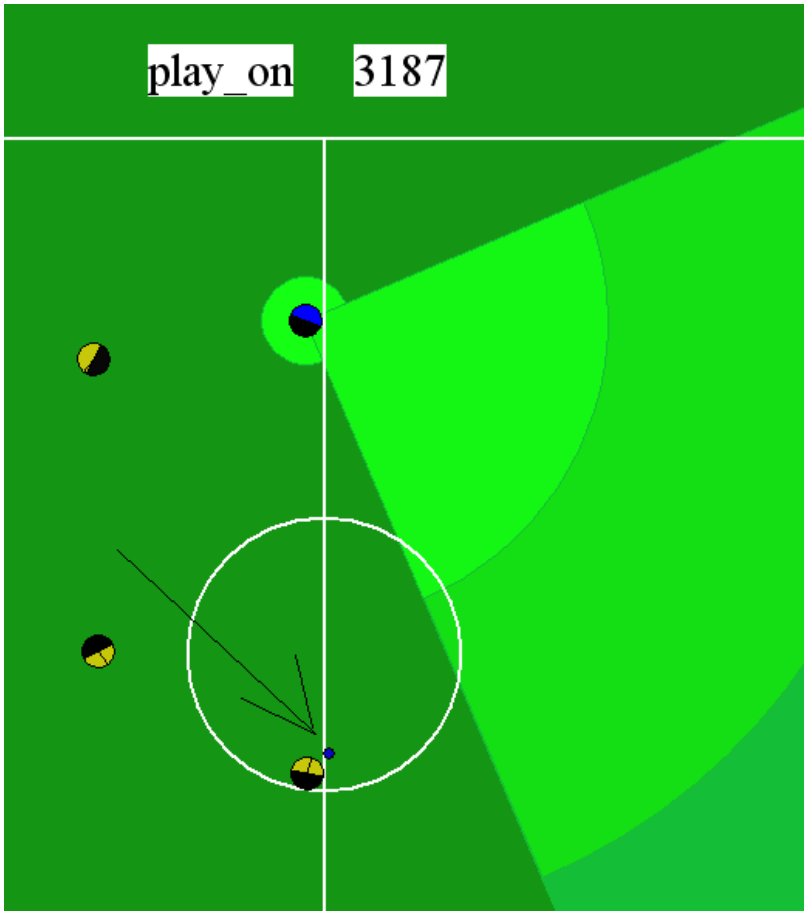
Na obrázku 26 vidno, že predtým ako server padne, nevie rozpársovať “LFAA1”, žiaľ táto chyba samotný pád serveru nespôsobuje. Pri testovaní bolo neskôr zistené, že táto chyba nespôsobuje okamžitý pád serveru. Simulácia, aj po tejto chybe, ešte dlho pokračuje. Z tohto výpisu vidno, že server pristupuje k pamäti, ktorá mu nepatrí. Logy, do ktorých server zapisuje prečo padol, neposkytujú žiadne relevantné informácie.

Obrázok 27 je tesne pred pádom servera. Lopta je nahraná zo smeru šípky a dalo by sa predpokladať, že pád servera by mohol byť spôsobený v strome, kedy hráč ešte nemá loptu. Tento predpoklad sa po mnohých testoch a modifikáciách podarilo podporiť. No chyba sa nedala odstrániť takým spôsobom, aby mal hráč zachovanú pôvodnú funkcionalitu.

Kolegovia z konkurenčného tímu prišli na to, že ich hráč síce beží na linuxovom RoboCup serveri bez pádu, no rovnako zhadzuje windows verziu serveru. Teda má rovnaký problém ako hráč UTTP. Podobne pôvodný UTTP hráč beží bez pádu na Linux RoboCup serveri. Tento problém sa

vyrieši tak, že fakultný RoboCup pobeží na Linux serveri a ani jeden hráč by nemal mať problém so zhadzovaním servera. UTTP hráč ešte musí byť dodatočne otestovaný na Linux serveri. Zaznamená chyba pádu servera bola odoslaná implementačnému tímu RoboCupu.

Počas hľadania dôvodu padania servera bola opravená jedna hrubá chyba, čo spôsobovala pád samotného hráča. Keď hráč nezaznamenal vo svojom okolí žiadneho nepriateľského hráča, tak vracala metóda hráča s indexom –1 a pritom v poli taký hráč nebol. Nový server by mal obsahovať rozsiahlejšie zmeny v týchto modeloch: dash model, stamina model, tacle model. Žiaľ rozsiahlejší popis týchto nových modelov, sa doteraz nepodarilo získať.



Obrázok 27 Pád serveru 13.0.2 záber z hry.

8 Overenie riešenia

Keďže hráč tímu Kukuričné deti vychádza z hráča tímu UTTP, rozhodli sme sa konfrontovať práve tieto dva tímy vo vzájomných zápasoch. Celkovo sa uskutočnilo desať polčasových stretnutí a štatistické údaje o ich priebehu môžeme vidieť v tabuľkách 2 a 3.

Na základe získaných štatistických údajov o priebehu hry, možno skonštatovať mierny progres v hraní. Hráč tímu Kukuričné deti dokázal zdolať súpera dvakrát, zatiaľ čo súper nezvíťazil ani raz. Vo väčšine zápasov (až 70%) sa viac ako 50% herného času hralo na súperovej polovici ihriska. V ostatných sledovaných ukazovateľoch sme nezaznamenali väčšie disproporcie.

Tabuľka 2 Štatistika Kukuričné deti**.**

Polčas	Počet strelených gólov	Počet striel na bránu	Čas hrania na súperovej strane ihriska (v %)	Úspešnosť zachytenia prihrávok (v %)	Úspešné prihrávky (v %)
1	0	1	51,24	50,87	42,27
2	0	0	46,12	50	48,78
3	0	3	52,43	55,1	44,56
4	1	2	51,11	54,17	44,9
5	0	1	54,17	45,88	52,22
6	0	2	53,59	52,88	41,94
7	0	0	52	51,81	52,22
8	0	0	55,24	46,05	48,65
9	1	1	45,71	53,4	49,53
10	0	0	49,39	60,4	42,72
Priemerné hodnoty	0,2	1,0	51,10	52,06	46,78

Tabuľka 3 Štatistika Kukuričné deti**.**

Polčas	Počet strelených gólov	Počet striel na bránu	Čas hrania na súperovej strane ihriska (v %)	Úspešnosť zachytenia prihrávok (v %)	Úspešné prihrávky (v %) (v %)
1	0	0	48,76	57,72	50,42
2	0	1	53,88	51,22	50
3	0	0	47,57	51	44,9
4	0	2	48,89	55,1	45,83
5	0	2	45,83	47,78	54,12
6	0	1	46,41	49,54	47,11
7	0	3	48	47,78	48,19
8	0	2	44,76	51,35	53,95
9	0	0	54,29	50,47	46,6
10	0	0	50,61	57,28	39,6
Priemerné hodnoty	0,0	1,1	48,90	51,92	48,07

Hráč Kukuričné deti bol ďalej externe testovaný tímom 12 Hráč. Tím 12 Hráč je ďalším tímom, ktorý sa venoval problematike RoboCup v rámci predmetu Tvorba softvérového systému v tíme 2009. Tím 12 Hráč nezaznamenal žiadne problémy s funkcionalitou nášho hráča.

9 Zhodnotenie

Zanalyzovali sme viaceré tímy, ktoré sa pravidelne umiestňujú na popredných miestach v súťažiach RoboCup. Po tejto analýze sme získali dostatočné vedomosti z implementácie rôznych typov správaní sa týchto hráčov. Výber hráča sme zúžili na dvoch fakultných. Oba tieto tímy mali svoje výhody, aj nevýhody. Nakoniec sme sa rozhodli pre hráča tímu *UTTP*. Tento tím síce minulý rok nevyhral, ale je modulárnejší, multiplatformový a integrovanie nových typov správania bude menej komplikované.

Analýza väčšieho množstva typov správaní nám poskytla pomerne široký výber rôznych typov správania sa hráčov, ktoré by sme mohli prebrať. V rámci prototypu sme z týchto prístupov implementovali tie najjednoduchšie. Ďalej sme dokázali implementovať niektoré vlastné vylepšenia, konkrétne komunikáciu medzi hráčmi alebo opravu vyčerpávania staminy obrancami. Táto práca na prototype nám ukázala niekoľko nedostatkov pôvodného hráča tímu UTTP, ktoré by sme chceli a čiastočne sme už aj odstránili, prípadne zlepšili.

Po hlbšom oboznámení sa so zdrojovými kódmi hráča sme ale dospeli k záveru, že prioritnejšou úlohou pre tento semester bude oprava jeho existujúcich správaní, ako integrácia nových. Rozšírili sme hráča o nový model sveta, ktorý predstavuje nový prístup k vnímaniu okolitých hráčov. Na základe tohto modelu sme realizovali prihrávanie si hráčov v hre. Ďalšou významnou zmenou bolo zlepšenie rozohrávania brankára. Odstránili sme statickú podobu tohto rozohrávania, čo významne znížili riziko inkasovania gólu pri vykopávaní lopty brankárom. Testovali sme nášho hráča na serveri 13, ktorý chceme použiť na najbližšom turnaji RoboCup na STU. Identifikovali sme problémy s naším hráčom na tomto serveri a eliminovali sme ich.

Celkovo sa ale integrácia vyššie opísaných zmien prejavila len miernym zlepšením hry hráča. Tento výsledok ale nepovažujeme za negatívny. Dokázali sme totiž integrovať vlastné moduly, založiť na nich kľúčové akcie hráča a pritom nestratiť pôvodnú kvalitu. Navyše tieto nové moduly predstavujú oveľa väčšie možnosti pri ďalšom vylepšovaní.

Našou snahou bude naďalej vylepšovať fakultného hráča *UTTP*, minimálne do turnaja RoboCup, ktorý sa uskutoční na STU 29 mája 2009.

9.1 Možné vylepšenia

V rámci projektu stále pracujeme na niekoľkých vylepšeniach hráča. Predovšetkým sa jedna o opravu formácií hráčov, aby sa celkovo formácia dynamicky premiestňovala s pohybom lopty do útočného pásma a späť. Momentálne je táto formácia statická.

Ďalšie vylepšenie sa týka obzerania sa hráča. Chceme ďalej vylepšovať toto obzerania sa, za cieľom mať loptu a situáciu v hre neustále pod kontrolou. Efektívne obzeranie sa je kľúčové pre už implementovanú nadstavbu modelu sveta.

Za účelom testovania modelu sveta hráča plánujeme vytvoriť vizualizačný systém, ktorý by zobrazoval aktuálne informácie o stave sveta pre každého jedného hráča počas zápasu.

Všetky tieto vyššie vymenované vylepšenia, s výnimkou vizualizačného systému, plánujeme dokončiť do turnaja RoboCup, ktorý sa uskutoční na STU v máji roku 2009.

Použitá literatúra

[1] Riedmiller, M., Gabel, T., Schulz, H.: Brainstormers 2D Public Source Code Release 2005. [Dátum posledného prístupu] 12. November 2008, Dostupné z <http://www.ni.uos.de/fileadmin/user_upload/publications/riedmiller.gabel.schulz.bs05publicrelease.pdf>,

[2] Sebastian, M.: OXSY 2006 Team Description. [Dátum citácie] 12.November 2008, Dostupné z <http://navid.alamati.googlepages.com/Oxsy2006TeamDescription.pdf>

[3] Flentge, F., Schneider, Ch., Wache, M.: Mainz Rolling Brains 2002. [Dátum posledného prístupu] 12. Novembra 2008, Dostupné z <www.informatik.uni-mainz.de/RollingBrains/MRB2002.ps>

[4] Nakayama, K., Ako, T., Suzuki, T., Takeuchi, I.: Team YowAI-2002. [Dátum posledného prístupu] 12. Novembra 2008, Dostupné z < http://ne.cs.uec.ac.jp/~koji/YowAI2002/YowAI2002-team-description.pdf >

[5] Kajsa, P., Kasan, M., Kontút, B., Kováč, D., Nepšinský, M., Sekereš, T.: RoboCup – Nové stratégie. [Dátum posledného prístupu] 12. Novembra 2008, Dostupné z <http://labss2.fiit.stuba.sk/TeamProject/2007/team06is-si/team06.pdf>

[6] Borženský, L., Brtáň, P., Kohut, J., Koperdák, M., Janov, V., Petráš, M.: Jahodoví Princovia Robocup S – Nové stratégie. [Dátum posledného prístupu] 12. Novembra 2008, Dostupné z < http://labss2.fiit.stuba.sk/TeamProject/2007/team16is-si/dokumenty/odovzdane2/Implementacia.pdf>

[7] Yao, J., Chen, J., Cai, Y., Li, L.: Architecture of TsinghuAeolus. In: *Proceedings of the 5th RoboCup Competitions and Conferences*, (2001), 491-494.

[8] Buttinger, S., Diedrich, M., Henning, L., Hoenemann, A., Huegelmeyer, P., Nie, A., Pegam, A., Rogowski, C., Rollinger, C., Steffens, T., Teiken, W.: The Dirty Dozen Team and Coach Decription. [Dátum posledného prístupu] 12. Novembra 2008, Dostupné z http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.23.7796>

[9] Fard, M., A., Ansari, H., M., Ildermi, A., Molavipour, S., Aledaghi, M., Seifi, F., Naghibzadeh, M.: Nexus 2D 2008 Team Decsription. [Dátum posledného prístupu] 12. Novembra 2008, Dostupné z http://nexus.um.ac.ir/Nexus-2d-2008.pdf>

Príloha A – Opis modelu sveta (World Model)

1 Opis modelu sveta

Táto časť opisuje model sveta (World Model) hráča Kukuričné deti. Svet uchováva informácie o objektoch v hre. Je základom pre korektné správanie sa hráčov. Táto kapitola opisuje jednotlivé triedy, ktoré tento svet tvoria, pričom taktiež opisuje volanie jednotlivých metód v rámci týchto tried.

V rámci hráča Kukuričné deti existujú tri hlavné typy podprojekty v rámci hlavného projektu a to:

- Coach (kouč)
- Goalie (brankár)
- Player (hráč)

Každý z týchto projektov predstavuje istý typ hráča v hre, ktorý sa dá logicky určiť podľa ich názvu. Tieto tri podprojekty majú niektoré moduly fyzicky spoločné, iné zase samostatné. V prípade modelu sveta sa jedná o nasledujúce moduly:

- *Player/Common/World* – tento modul je spoločný pre hráča typu Coach, Goalie aj Player.
- *Player/Player/World(Player)* – tento modul je spoločný pre hráčov typu Goalie a Player. Coach tento modul logicky nemá, keďže jeho postavenie v rámci hry je odlišné.

2 Základné rozdelenie

Táto časť opisuje základné rozdelenie modelu sveta. Úplným základom modelu sveta je modul *Player/Common/World.* Tento modul poskytuje základné štruktúry pre prácu s analytickou geometriou:

- **Angle.h** – trieda pre prácu s uhlami (v radiánoch a v stupňoch).
- **ParamStructs.h** – zoznam parametrov použitých v rámci modelu sveta (napr. *ball_decay*, *ball_size*, atď.).
- **Point.h** – reprezentácia bodu v ortogonálnom súradnicovom systéme.
- **Vector.h** - reprezentácia vektora v ortogonálnom súradnicovom systéme.

Modul *Player/Player/World(Player)* obsahuje informácie o hre na základnej úrovni, teda informácie o lopte, hráčoch, stave sveta, atď. Nasledujúci zoznam opisuje, ktoré informácie sú kde uchované:

- **Informácie o aktuálnej pozícii lopty (naposledy videnej pozície) počas priebehu hry** – trieda *Player/Player/World(Player)/Ball.h:*
- **Informácie o samotnom hráčovi (ten, ktorý práve hrá) v rámci hry** – trieda *Player/Player/World(Player)/Myself.h:*
- **Informácie o okolitých hráčoch v hre (súperi, spoluhráči, neznámi hráči)** – tieto informácie sú uložené v poli *friends* (spoluhráči) a v poli *enemies* (súperi, neznámi) v triede *Player/Player/World(Player)/World.h*. Tieto polia - *friends* a *enemies* sú typu *Player* triedy *Player/Player/World(Player)/Player.h*
- **Informácie o priebehu stavu hry (vzájomné skóre, čas v hre, mód hry)** – trieda *Player/Player/World(Player)/State.h*

3 Opis hlavných tried modelu sveta

Táto časť opisuje funkciu hlavných tried v rámci modelu sveta, teda čo jednotlivé triedy uchovávajú za informácie.

Trieda Ball

Umiestnenie:
Player/Player/World/Ball.cpp
Player/Player/World/Ball.h

Trieda *Ball* uchováva informácie o stave lopty v hre a to konkrétne nasledujúce údaje:

- Pozíciu lopty(poslednú známu)
- Čas, kedy sme loptu naposledy videli
- Informácia o tom, či sú údaje správne

Prístup k tejto štruktúre z ľubovoľného modulu sveta, ktorý dedí od triedy *WorldObject*:

```
this->world->ball
```

Trieda Enemy

Umiestnenie:
Player/Player/World/Enemy.cpp
Player/Player/World/Enemy.h

Trieda implementovaná tímom Kukuričné deti pre rozšírenie modelu sveta o nový spôsob vnímania okolia hráčom. Podrobnejšie informácie k tejto triede sú uvedené v kapitole 7.3 tohto dokumentu.

Trieda Movable

Umiestnenie:
Player/Player/World/Movable.cpp
Player/Player/World/Movable.h

Trieda *Movable* predstavuje implementáciu všeobecne pohybujúceho sa objektu v hre. Od tejto triedy dedia hlavne nasledujúce triedy:

- Player
- Ball
- *Myself* (Trieda *Myself* dedí od triedy *Player*, ktorá následne dedí od triedy *Movable)*

Táto metóda uchováva hlavne bodovú pozíciu hráča a smer jeho pohybu.
Point pos – bodová pozícia hráča
Vector vel – smer pohybujúceho sa objektu

Trieda Myself

Umiestnenie:
Player/Player/World/Myself.cpp
Player/Player/World/Myself.h

Trieda *Myself* predstavuje objekt hráča, ktorý momentálne hrá zápas. Táto trieda logicky dedí od triedy *Player*, pričom doplňuje ďalšie informácie, ktoré o svojom hráčovi uchovávame. Sú to informácie ako hodnota energie hráča, šírky pohľadu, atď.

Trieda Params

Umiestnenie:
Player/Player/World/Params.cpp
Player/Player/World/Params.h

Trieda *Params* predstavuje parametre servera v modeli sveta. Táto trieda obsahuje ukazovateľe na tri typy parametrov:

SERVER_PARAMS
PLAYER_PARAMS

`PLAYER_TYPE`

Pri bližšie informácie o parametroch týchto troch množín je nutné pozrieť zdrojový kód: `Player/Player/Common/World/ParamStructs.h`

Trieda Player

Umiestnenie:
Player/Player/World/Player.cpp
Player/Player/World/Player.h

Trieda *Player* predstavuje objekt reprezentujúci hráča v hre. Keďže hráč je pohybujúci sa objekt, táto trieda dedí od triedy *Movable*. Trieda *Player* uchováva základné informácie, ako sú číslo dresu hráča, natočenie hlavy hráča, natočenie tela hráča, atď. Jedná sa o informácie, ktoré sme schopný rozoznať pozorovaním aj na iných hráčoch, preto je jej inštancia použitá aj pri ukladaní informácií o spoluhráčoch a protihráčoch v hre.

Celkovo je táto trieda použitá ako štruktúra pre uchovávanie informácii o nepriateľovi. Poskytuje tak hlavne *get* a *set* metódy pre premenné, ktoré obsahuje.

Trieda State

Umiestnenie:
Player/Player/World/State.cpp
Player/Player/World/State.h

Trieda *State* zapuzdruje informácie o stave hry. Je tu uložená hodnota simulačného času, mód hry, ako aj hodnota skóre oboch tímov v hre.

Trieda Variables

Umiestnenie:
Player/Player/World/Variables.cpp
Player/Player/World/Variables.h

Trieda *Variables* obsahuje premenné, ktoré sú pravdepodobne využívané koučom. V budúcnosti je potrebné zistenie presného významu tejto triedy. Obsahuje totiž parametre, ktoré sú využívané aj v správaní sa hráčov počas hry.

Trieda World

Umiestnenie:
Player/Player/World/World.cpp
Player/Player/World/World.h

Trieda World je hlavnou triedou modelu sveta. Zapuzdruje informácie o okolitom prostredí hráča a tým vytvára model sveta. Svet hráča pozostáva zo samotného hráča, ostatných hráčov, lopty, premenných, atď. Táto trieda preto zastrešuje všetky ostatné triedy vymenované v tejto kapitole, teda:

- class Ball
- class Player
- class Myself
- class Enemy
- class State
- class Params
- class Variables

Dôležité štruktúry tejto triedy sú nasledujúce:

`DynArray<Player, 32> friends` – pole spoluhráčov, ktorých hráč vidí, prípadne videl
`DynArray<Player, 32> enemies` – pole protihráčov, ktorých hráč vidí, prípadne videl

Táto triedy obsahuje aj základné metódy, ktoré aktualizujú model sveta. Jedná sa o nasledujúce metódy:

```
void World::ProcessInfo(const SENSE_BODY* sb, const VISUAL_INFO* vi)
void World::RecalculateEnemies(const SENSE_BODY *sb, const VISUAL_INFO *vi)
void World::RecalculateFriends(const SENSE_BODY* sb, const VISUAL_INFO* vi)
```

Trieda WorldObject

Umiestnenie:
Player/Player/World/WorldObject.h

Trieda *WorldObject* predstavuje triedu na najvyššej úrovni v hierarchii sveta. Od tejto triedy teda dedia ostatné triedy sveta, konkrétne:

- Enemy
- Movable
- Params
- State
- Variables

Trieda *WorldObject* celkovo predstavuje inštanciu triedy *World*. Obsahuje dve základné metódy:

`World* GetWorld() const` - Vracia ukazovateľ na svet, ktorému patrí tento objekt
`void SetWorld(World* w)` - Nastaví ukazovateľ na svet, ktorému patrí tento objekt

Príloha B – Opis správaní hráča

V tejto prílohe uvádzame prevzatú časť z dokumentácie UTTP hráča, za účelom vytvorenia ucelenej a prehľadnej technickej dokumentácie k hráčovi Kukuričné deti, ktorý z hráča UTTP vychádza.

1 Identifikované správania

Tento dokument obsahuje správania, ktoré sa nám podarilo identifikovať v zdrojových kódoch tímu Loptoši. Ku každému správaniu sa uvádza algoritmus správania, závislosť od iných objektov a lokalizácia opisu správania v dokumentáciách tímov.

1.1 Elementárne správania

Za elementárne správania považujeme správania, ktoré sa už ďalej nezjemňujú, ale priamo vyžadujú zaslanie správ na server.

Zmena pohľadu

Algoritmus:

Zmena pohľadu hráča sa uskutočňuje v metóde PlayerSkills::NeckSynchronization() pomocou metódy CommandsInterface::ChangeView().

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`CommandsInterface::ChangeView()`**

`PlayerSkills::TurnNeckToAux()`

Lokalizácia:

Správanie Zmena pohľadu je lokalizované v dokumentácii tímu StjupidDox, ktorý toto správanie navrhol avšak neimplementoval.

Posielanie správ

Algoritmus:

Posielanie správ sa uskutočňuje v metóde PlayerSkills::sendInfo() pomocou metódy CommandsInterface::ChangeView(). V tejto metóde sa aj správa vytvorí.

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`CommandsInterface::Say()`**

Lokalizácia:

Správanie Posielanie správ patrí k elementárnym správaniam , ktoré poskytuje server.

Otočenie krku

Algoritmus:

Zmena natočenia krku sa uskutočňuje v metóde PlayerSkills::TurnNeckToAux()pomocou metódy CommandsInterface::TurnNeck().

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`CommandsInterface::TurnNeck()`**

Lokalizácia:

Správanie Otočenie krku je lokalizované v dokumentácii tímu StjupidDox, ktorý toto správanie ako prvý implementoval a tímu FIITBA, ktorý toto správanie zmenili na otáčanie krku.

Otočenie

Algoritmus:

Natočenie hráča sa uskutočňuje v metóde PlayerSkills::TurnTo() pomocou metódy CommandsInterface::Turn().

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`CommandsInterface::Turn()`**

Lokalizácia:

Správanie Otočenie patrí k elementárnym správaniam , ktoré poskytuje server.

Presun

Algoritmus:
Presun hráča sa uskutočňuje v metóde PlayerSkills::MoveTo() pomocou metódy CommandsInterface::Move().

Závislosti:
funkcie: **`CommandsInterface::Move()`**

Lokalizácia:

Správanie Presun patrí k elementárnym správaniam , ktoré poskytuje server.

Chytenie lopty

Algoritmus:
Chytenie lopty sa uskutočňuje v metóde PlayerSkills::CatchBall() pomocou metódy CommandsInterface::Catch().

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`CommandsInterface::Catch()`**

Lokalizácia:

Správanie Chytenie lopty patrí k elementárnym správaniam , ktoré poskytuje server.

Šmýkačka

Algoritmus:
Šmýkačka sa uskutočňuje v metóde PlayerSkills::Tackle() pomocou metódy CommandsInterface::Tackle().

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`CommandsInterface::Tackle()`**

Lokalizácia:

Správanie Šmýkačka patrí k elementárnym správaniam , ktoré poskytuje server.

Kopnutie

Algoritmus:

Kopnutie do lopty sa uskutočňuje v metóde PlayerSkills::DumbKick() pomocou metódy CommandsInterface::Kick().

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`CommandsInterface::Kick()`**

`PlayerSkills::TurnTo()`

Lokalizácia:

Správanie Kopnutie je lokalizované v dokumentácii tímu StjupidDox, neskôr sa pridalo kopnutie na bránu. Loptoši si však spravili vlastné správanie „Strela na bránku“.

Blízke kopnutie

Algoritmus:

Blízke kopnutie lopty sa uskutočňuje v metóde PlayerSkill::NearKick(). Najprv sa zistí, či je daný hráč dostatočne blízko k lopte. Potom zistíme potrebnú veľkosť rýchlosti kopnutia alebo či je potrebné najprv loptu zastaviť. Nakoniec samotné kopnutie do lopty vykonáme pomocou metódy CommandsInterface::Kick().

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`CommandsInterface::Kick()`**

Lokalizácia:

Správanie Blízke kopnutie je lokalizované v dokumentácii tímu Roztoče, s týmto správaním úzko súvisí správanie dobehnutie lopty – NearKick.

1.2 Správanie hráča s loptou

Správanie s loptou sa rozdeľuje na základe toho, či daný hráč sa nachádza v hernom móde alebo je určený na rozohrávanie. Toto rozdelenie závisí od návratovej hodnoty metódy PlayerTactics::ShouldKickIn() pre každého hráča. Hráč určený na rozohrávanie sa vyberie na základe najbližšej domovskej pozícii hráča k miestu rozohrávania. Hráč určený na rozohrávanie realizuje inštrukcie v bloku za podmienkou if(ShouldKickIn()), ostatní hráči realizujú inštrukcie bloku za else.

Prihrávanie

Algoritmus:

Prihrávanie sa vykonáva v metóde PlayerSkills::PassBallTo(). Najprv sa upraví rýchlosť prihrávky a potom sa vypočíta cieľová pozícia prihrávky. Prihranie sa vykoná v metóde PlayerSkills::NearKick() (identifikované ako správanie blízke kopnutie). Po prihrávke hráč natočí hlavu na cieľ prihrávky pomocou metódy PlayerSkills::TurnNeckTo() (identifikované ako správanie pozeranie)

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::NearKick()`**

`PlayerSkills::TurnNeckTo()`

Lokalizácia:

Správanie tímové nahrávanie je lokalizované v dokumentácii tímu FIITBA.

Výstrel na bránu

Algoritmus:

Toto správanie sa realizuje v metóde PlayerSkills::LPT_KickToGoal(). Vypočíta bod, do ktorého hráč vystrelí plnou silou. Strela bude smerovať k ľavej alebo pravej tyči podľa toho, ku ktorej tyči je lopta bližšie. Samotný výstrel na bránu zabezpečuje metóda PlayerSkills::DumbKick(), ktorá je identifikovaná ako správanie kopnutie.

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::DumbKick()`**

Lokalizácia:

Správanie Výstrel na bránu je lokalizované v dokumentácii tímu Loptoši.

Pozeranie

Algoritmus:

Toto správanie je implementované v metóde PlayerSkills::TurnNeckTo(). Hráč sa pozrie na danú pozíciu pomocou metódy PlayerSkills::NeckSynchronization() (identifikované ako zmena pohľadu) iba ak je vypnutá synchronizácia krku s pohľadom.

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::NeckSynchronization()`**

Lokalizácia:

Správanie prihrávky do behu je lokalizované v dokumentácii tímu Roztoče. Je možné nájsť pod názvom „sledovanie lopty(stratégia sledovania ihriska)“.

Správanie rozohrávajúceho hráča

Hra je pozastavená a čaká sa na našu rozohrávku (resp. vhadzovanie). Hráč, ktorý je určený na rozohrávku sa poobzerá okolo seba (metóda PlayerTactics::ExamineSurroundings()), a tým získa informácie z okolitého sveta. Toto správanie je identifikované ako spoznávanie okolia. Po spoznaní okolia nasleduje vykonanie inštrukcií prislúchajúcich danému stavu sveta. Ak sa nachádzame v hracom móde *PM_CornerKick_Our*, realizuje sa správanie identifikované ako rohový kop. Ak sa nachádzame v hracom móde *PM_FreeKick_Our* alebo *PM_IndFreeKick_Our*, realizuje sa správanie identifikované ako voľný kop, individuálny voľný kop. Ak sa nachádzame v hracom móde *PM_KickIn_Our*, realizuje sa správanie identifikované ako vhadzovanie. Ak sa nachádzame v hracom móde *PM_KickOff_Our alebo PM_AfterGoal_Their*, realizuje sa správanie identifikované ako zahrávanie. Ak sa nachádzame v hracom móde *PM_PenaltyKick_Our*, realizuje sa správanie identifikované ako pokutový kop.

Náš rohový kop

Algoritmus:

Hráč, ktorý ide uskutočniť rohový kop sa v prvom cykle natočí na bod 25,0 (stred súperovej polovice) pomocou metódy PlayerSkills::TurnTo() (identifikované ako správanie otočenie). Ďalších 29 cyklov hráč čaká na umiestnenie spoluhráčov. Nasleduje rozohranie spoluhráčovi (identifikované ako správanie rozohranie) pomocou metódy PlayerTactics::StartPlayOnByPassToOurPlayer(), ktorá má návratovú hodnotu true ak sa rozohralo spoluhráčovi, inak false.

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::TurnTo()`**

`PlayerTactics::StartPlayOnByPassToOurPlayer()`

Lokalizácia:

Správanie rozohrávanie štandardných situácií a zahrávanie zo stredu je lokalizované v dokumentácii tímu FIITBA. Na toto nadviazal tím Loptoši a vytvorili správanie rohový kop, aut, ktoré je teda lokalizované v dokumentácii tímu Loptoši.

Náš voľný kop, Náš individuálny voľný kop

Algoritmus:

Hráč, ktorý ide uskutočniť voľný alebo individuálny voľný kop sa v prvom cykle natočí na bod 52.5,0 (stred súperovej bránky) pomocou metódy PlayerSkills::TurnTo() (identifikované ako správanie otočenie). Ďalších 29 cyklov hráč čaká na umiestnenie spoluhráčov. Nasleduje rozohranie spoluhráčovi (identifikované ako správanie rozohranie) pomocou metódy PlayerTactics::StartPlayOnByPassToOurPlayer(), ktorá má návratovú hodnotu true ak sa rozohralo spoluhráčovi, inak false.

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::TurnTo()`**

`PlayerTactics::StartPlayOnByPassToOurPlayer()`

Lokalizácia:

Správanie voľný kop je lokalizované v dokumentácii tímu FIITBA.

Naše vhadzovanie

Algoritmus:

Hráč, ktorý ide uskutočniť vhadzovanie si najprv v prvom cykle zistí, či sa nachádza v hornej alebo dolnej časti ihriska. Ak sa nachádza v hornej časti ihriska, natočí sa na x,y+15 (kde x a y sú súradnice hráča, ktorý ide vhadzovať), ak sa nachádza v dolnej časti ihriska, tak sa hráč natočí na x,y-15. Natočenie sa uskutoční pomocou metódy PlayerSkills::TurnTo() (identifikované ako správanie otočenie). Ďalších 29 cyklov hráč

čaká na umiestnenie spoluhráčov. Nasleduje rozohranie spoluhráčovi (identifikované ako správanie rozohranie) pomocou metódy PlayerTactics::StartPlayOnByPassToOurPlayer(), ktorá má návratovú hodnotu true ak sa rozohralo spoluhráčovi, inak false.

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::TurnTo()`**

`PlayerTactics::StartPlayOnByPassToOurPlayer()`

Lokalizácia:

Správanie rozohrávanie štandardných situácií a zahrávanie zo stredu je lokalizované v dokumentácii tímu FIITBA. Na toto nadviazal tím Loptoši a vytvorili správanie rohový kop, aut, ktoré je teda lokalizované v dokumentácii tímu Loptoši.

Naše zahrávanie

Algoritmus:
Hráč, ktorý je určený na zahrávanie najprv pomocou metódy CEvaluatingPlayerList::GetKickOffPlayer() získa hráča, ktorý je vhodný pre prihrávku. Ak takýto hráč existuje, tak zahrávajúci hráč mu zahrá loptu silou 1, inak zahrá loptu na -25,0 (stred obranného pásma) silou 0.7. Samotné zahranie (identifikované ako správanie prihrávanie) sa uskutoční pomocou metódy PlayerSkills::PassBallTo().

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

`CEvaluatingPlayer` (hráč z vizuálnej informácie iného hráča)

`CEvaluatingPlayerList` (zoznam hráčov z vizuálnej informácie hráča)

funkcie: **`PlayerSkills::TurnTo()`**

`PlayerSkills::PassBallTo()`

Lokalizácia:

Správanie rozohrávanie štandardných situácií a zahrávanie zo stredu je lokalizované v dokumentácii tímu FIITBA. Na toto nadviazal tím Loptoši a vytvorili správanie rohový kop, aut, ktoré je teda lokalizované v dokumentácii tímu Loptoši.

Náš pokutový kop

Algoritmus:

Pokutový kop sa uskutoční pomocou metódy PlayerSkills::LPT_KickToGoal() (identifikované ako správanie výstrel na bránu).

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::LPT_KickToGoal()`**

Lokalizácia:

Správanie pokutový kop je lokalizované v dokumentácii tímu FIITBA.

Naše rozohranie

Algoritmus:

Rozohranie sa uskutočňuje v metóde PlayerTactics::StartPlayOnByPassToOurPlayer(). Na začiatku sa vypočíta vhodná pozícia v okolí hráča pre normálnu prihrávku pomocou metódy CEvaluatingPlayerList::CalculateNormalPassPosition(). Ak existuje takáto pozícia vypočítame potrebnú silu prihrávky a prihráme pomocou metódy PlayerSkills::PassBallTo() (identifikované ako správanie prihrávanie). Ak takáto pozícia neexistuje, prihráme pomocou metódy PlayerSkills::PassBallTo() (identifikované ako správanie prihrávanie) na pozíciu 40,0 (útočné pásmo).

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

`CEvaluatingPlayerList` (zoznam hráčov z vizuálnej informácie hráča)

`CEvaluatingPlayer` (hráč z vizuálnej informácie iného hráča)

`CEvaluatingPanesMap` (vyhodnocovacia mapa)

funkcie: **`CEvaluatingPlayerList::CalculateNormalPassPosition`**
`PlayerSkills::PassBallTo()`

Lokalizácia:

Správanie Rozohranie je lokalizované v dokumentácii tímu FIITBA.

1.2.1 Správanie hráča s loptou počas herného módu

Tieto správania sa využívajú iba v prípade, ak má hráč loptu a zápas je v hernom móde, ktorý nezahŕňa rôzne štandardné situácie ako sú rôzne kopy na bránu a vhadzovanie.

Strela na bránku

Algoritmus:

Rozhodovania o strele zabezpečuje metóda PlayerTactics::ShouldKickOnGoal(). Ak má hráč loptu, prioritne zvažuje možnosť vystreliť na súperovu bránu. Hráč určite vystrelí na bránu ak je vzdialenosť lopty od brány menšia ako 17. Naopak, nepokúsi sa vystreliť, ak je vzdialenosť lopty od brány väčšia ako 25. Ak je aktuálna vzdialenosť medzi 17 a 25, tak hráč zvažuje na základe ďalších kritérií. Jedno z kritérií je informácia o súperovom brankárovi. Ak hráč nevidí brankára, tak nevystrelí na bránku okrem prípadu, kedy je bránka prázdna, čo sa zistí podľa bodu, na ktorý sa hráč pozerá. Ak je viditeľný aj súperov brankár, tak sa hráč snaží vypočítať, či je možné streliť gól, podľa pozície súperovho brankára a ostatných hráčov.

Vypočítanie bodu, do ktorého hráč vystrelí plnou silou zabezpečuje metóda PlayerSkills::LPT_KickToGoal() (identifikované ako správanie výstrel na bránu).

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::LPT_KickToGoal()`**

`PlayerTactics::ShouldKickOnGoal()`

Lokalizácia:

Roztoče urobili strieľanie gólov - nájdenie maximálneho uhla pri strele na bránku. V Tíme FC Farmárov došlo k zmene vzdialenosti strieľania na bránu.

Prekopnutie ofsajdovej línie

Algoritmus:

Prekopnutie zabezpečuje metóda PlayerSkills::OffsideLineBreach(), ktorá využíva základné správania prihrávanie a pozeranie. Hráč sa pokúsi prekopnúť ofsajdovú líniu ak je od nej vzdialený maximálne 4,75 a nenachádza sa príliš hlboko v súperovom obrannom pásme. Toto správanie môže realizovať jedine hráč s číslom 8 alebo viac, čiže iba ofenzívny hráč.

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

`CEvaluatingPlayerList` (zoznam hráčov z vizuálnej informácie hráča)

`CEvaluatingPlayer` (hráč z vizuálnej informácie iného hráča)

`CEvaluatingPanesMap` (vyhodnocovacia mapa)

funkcie: **`PlayerSkills::OffsideLineBreach()`**

`CEvaluatingPlayerList::CalculateOptimalDribblePointAux`

PlayerSkills::PassBallTo()

PlayerSkills::TurnNeckTo()

Lokalizácia:

Zohľadňovanie ofsajdov a prekopnutie ofsajdovej línie pridali Roztoče. Ďalší postup : StjupidDox zmenili, FC Farmári zlepšili, FIITBA zmenili.

Driblovanie

Algoritmus:

Rozhodnutie o bode driblovania zabezpečuje metóda PlayerTactics::ManageDribblePoint(), ktorá vypočíta súradnice bodu, do ktorého hráč bude driblovať a vráti TRUE alebo vráti FALSE, ak nie je vhodne driblovať. Pri rozhodovaní hráča s loptou sa zvažuje driblovanie na troch miestach a ak nie je vhodné driblovať, tak sa volí správanie strategická prihrávka. Samotné driblovanie na vypočítaný bod vykonáva metóda PlayerSkills::DribbleTo2(). Na záver ak je hráč bližšie maximálne 35 metrov od súperovej brány, pokúsi sa pozrieť na súperovu bránu pomocou otočenia krku. Na obrázku (Obrázok A-1) je zobrazený stavový diagram pre driblovanie.

Závislosti:

objekty: **world** (zisťovanie informácií o pozícií, o čísle atď.)

CEvaluatingPlayerList (zoznam hráčov z vizuálnej informácie hráča)

CEvaluatingPlayer (hráč z vizuálnej informácie iného hráča)

CEvaluatingPanesMap (vyhodnocovacia mapa)

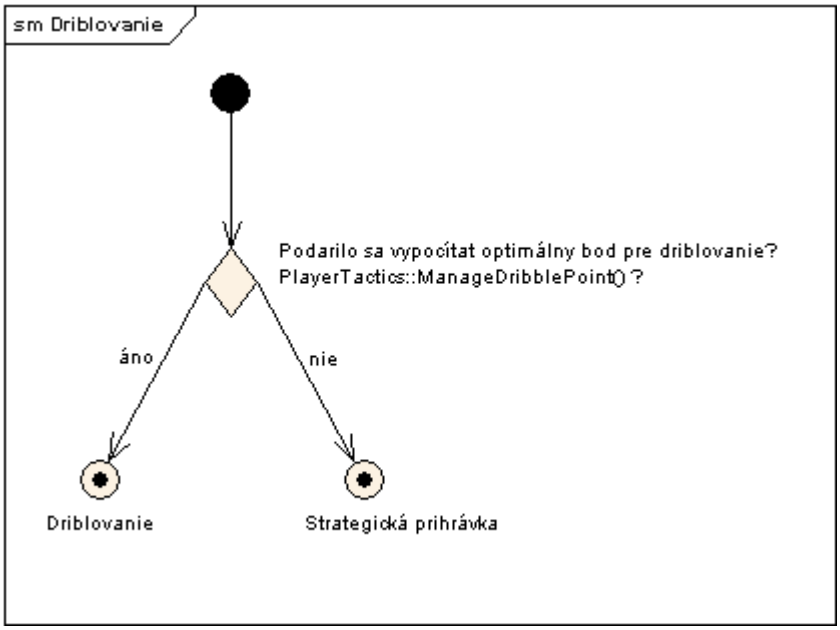
funkcie: **PlayerSkills::DribbleTo2()**

PlayerTactics::DistanceToGoal()

PlayerSkills::TurnNeckTo()

Lokalizácia:

Prvé driblovanie vytvoril tím Roztoče. Toto bolo zmenené tímom StjupidDox, konkrétne dosiahli rýchlejší pohyb s loptou na úkor jej presného ovládania. Driblovanie ďalej vylepšovali FC Farmári a takisto FIITBA.



Obrázok A-1 Stavový diagram pre driblovanie

Prihrávka1

Algoritmus:

Toto správanie sa využije ak sa podarí nájsť vhodnú pozíciu v okolí hráča pre prihrávku pomocou funkcie CEvaluatingPlayerList::CalculateNormalPassPosition() a nie je vhodné driblovať, pretože v okolí hráča sa nachádza súper (zistenie pomocou metódy CEvaluatingPlayerList::NoEnemiesAround()) alebo hráč je ďaleko od súperovej bránky.

Prihráva sa do bodu, ktorý vypočítala vyššie spomenutá metóda a sila sa určí podľa vzdialenosti. Prihranie sa uskutoční pomocou metódy PlayerSkills::PassBallTo() (identifikované ako správanie prihrávanie).

Závislosti:

objekty: **`world`** (zistovanie informácií o pozícií, o čísle atď.)

`CEvaluatingPlayerList` (zoznam hráčov z vizuálnej informácie hráča)

`CEvaluatingPlayer` (hráč z vizuálnej informácie iného hráča)

`CEvaluatingPanesMap` (vyhodnocovacia mapa)

funkcie: **`CEvaluatingPlayerList::NoEnemiesAround()`**

`PlayerSkills::PassBallTo()`

Lokalizácia:

Roztoče urobili nahrávanie - používanie prihrávky do behu. FC Farmári urobili zlepšenie nahrávania - rozhodovanie na základe neurónovej siete. Tímové nahrávanie vytvoril tím FIITBA.

Prihrávka2

Algoritmus:

Ak sa nepodarilo nájsť vhodnú pozíciu v okolí hráča pre prihrávku pomocou funkcie CEvaluatingPlayerList::CalculateNormalPassPosition() a strieľať nie je vhodné, pretože brankár je veľmi blízko (bližšie ako 5), tak sa volí Prihrávka2, ktorej parametre sa určia pomocou pozície hráča a pozície brankára. Prihranie sa uskutoční pomocou metódy PlayerSkills::PassBallTo() (identifikované ako správanie prihrávanie).

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

`CEvaluatingPlayerList` (zoznam hráčov z vizuálnej informácie hráča)

CEvaluatingPlayer (hráč z vizuálnej informácie iného hráča)

CEvaluatingPanesMap (vyhodnocovacia mapa)

funkcie: **PlayerSkills::PassBallTo()**

Lokalizácia:
Roztoče urobili nahrávanie - používanie prihrávky do behu. FC Farmári urobili zlepšenie nahrávania - rozhodovanie na základe neurónovej siete. Tímové nahrávanie vytvoril tím FIITBA.

Prihrávka3

Algoritmus:
Ak neboli splnené podmienky pre vykonanie Prihrávky2, tak sa zvažuje použitie Prihrávky3. Toto správanie sa použije, ak je hráč hlboko v pásme súpera a v tomto pásme sú aj nejaký jeho spoluhráči. Zistenie spoluhráčov v pásme hráča sa urobí pomocou metódy PlayerSkills::IsPointInRectArea(). Prihranie sa uskutoční pomocou metódy PlayerSkills::PassBallTo() (identifikované ako správanie prihrávanie).

Závislosti:
objekty: **world** (zisťovanie informácií o pozícií, o čísle atď.)

CEvaluatingPlayer (hráč z vizuálnej informácie iného hráča)

CEvaluatingPanesMap (vyhodnocovacia mapa)

funkcie: **PlayerSkills::PassBallTo()**

PlayerSkills::IsPointInRectArea()

CEvaluatingPlayerList::OurPlayersInEnemyPenaltyArea()

Lokalizácia:
Roztoče urobili nahrávanie - používanie prihrávky do behu. FC Farmári urobili zlepšenie nahrávania - rozhodovanie na základe neurónovej siete. Tímové nahrávanie vytvoril tím FIITBA.

Predkopnutie lopty

Algoritmus:
Vykonáva metóda PlayerTactics::OutPlayOpponent(). Hráč si predkopne loptu vľavo alebo vpravo od súpera. Toto je posledný variant, ktorý hráč vykoná vždy, ak nevykonal žiadnu inú akciu s loptou. Používa základné správania: prihrávanie, pozeranie.

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerTactics::GetNearestEnemy()`**

`PlayerSkills::PassBallTo()`

Lokalizácia:
Predkopnutie lopty pridal tím FIITBA

Strategická prihrávka

Algoritmus:
Hráč sa pokúsi prihrať na strategickú pozíciu iba ak najbližší súper je vzdialený o viac ako 1,5. Smer tejto prihrávky nie je závislý od pozície žiadnych hráčov a ak je lopta ďaleko od brány, tak hráč kopne celou silou dopredu inak prihrá loptu na náhodnú pozíciu dáždnika pomocou správania prihrávka na náhodnú pozíciu dáždnika.

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::PassToStrategicPosition()`**

`PlayerSkills::DumbKick()`

`PlayerSkills::PassToRandomUmbrellaPosition()`

Lokalizácia:
Práve toto správanie bolo nájdené v dokumentácii FIITBA, ktorý toto správanie aj implementoval.

Prihrávka na náhodnú pozíciu dáždnika

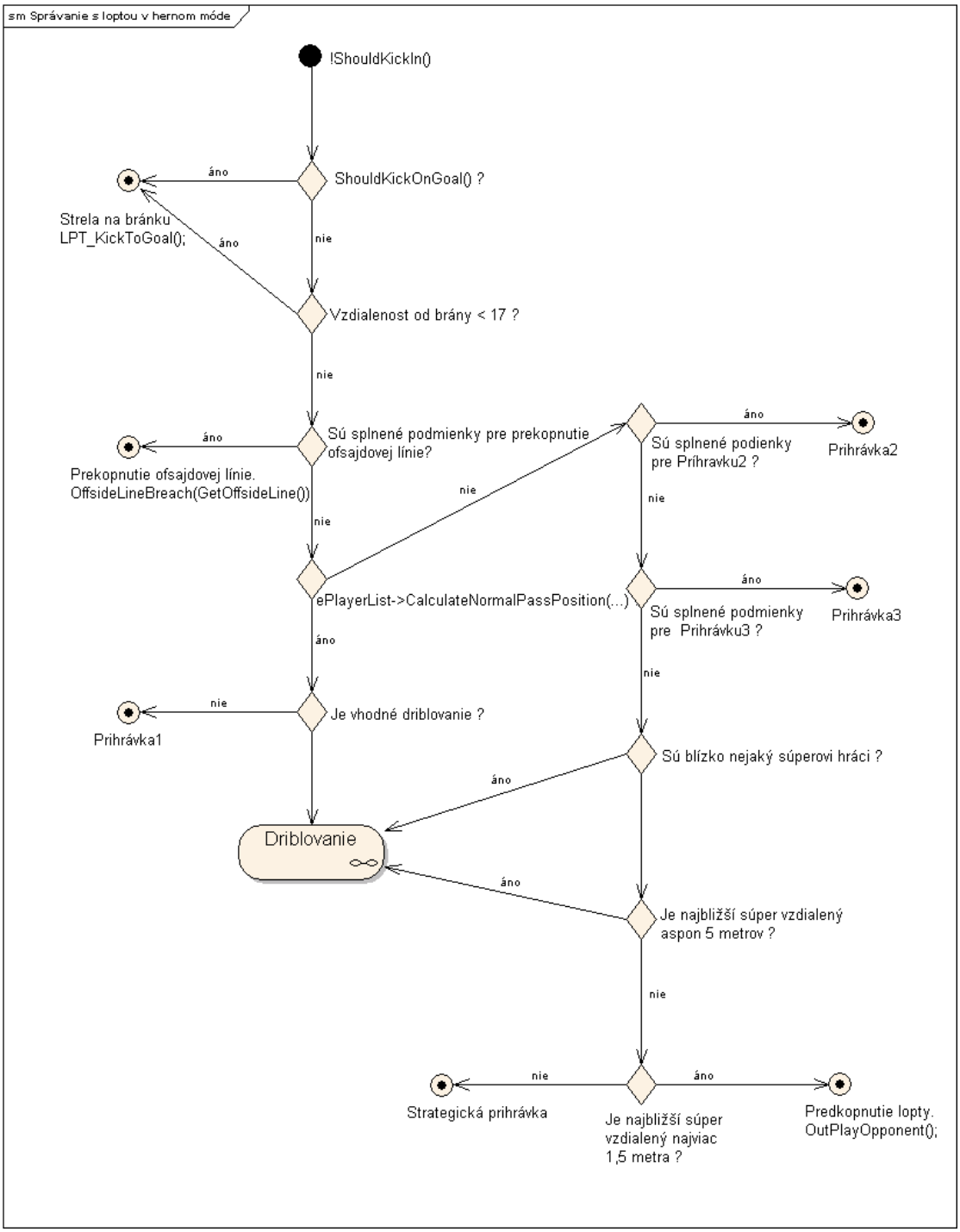
Algoritmus:
Zabezpečuje metóda PlayerTactics::PassToRandomUmbrellaPosition(), náhodne sa vyberie jeden bod z poľa umbrellaPositions a do tohto bodu hráč vystrelí pomocou metódy PlayerSkills::DumpKick (identifikované ako správanie kopnutie).

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::DumbKick()`**

Lokalizácia:
Práve toto správanie bolo nájdené v dokumentácii FIITBA, ktorý toto správanie aj implementoval.

Na obrázku (Obrázok A-2) je zobrazený kompletný stavový diagram pre hráča s loptou v hernom móde.



Obrázok A-2 Stavový diagram správania hráča čakajúceho na rozohrávku

1.3 Správanie hráča bez lopty

Správanie hráča bez lopty sa uskutočňuje v jednotlivých if-else blokoch na základe hracieho módu. Ak sa hráč nachádza v hernom móde *PM_CornerKick_Our,* vykoná sa správanie identifikované ako náš rohový kop. Ak sa hráč nachádza v hernom móde *PM_FreeKick_Our* alebo *PM_IndFreeKick_Our,* vykoná sa správanie identifikované ako náš voľný kop, náš individuálny voľný kop. Ak sa hráč nachádza v hernom móde *PM_GoalKick_Our,* vykoná sa správanie identifikované ako naša „päťka“. Ak sa hráč nachádza v hernom móde *PM_KickIn_Our,* vykoná sa správanie identifikované ako naše vhadzovanie. Ak sa hráč nachádza v hernom móde *PM_KickOff_Our,* vykoná sa správanie identifikované ako naše zahrávanie. Ak sa hráč nachádza v hernom móde *PM_PenaltyKick_Our,* vykoná sa správanie identifikované ako náš pokutový kop. Ak sa hráč nachádza v hernom móde *PM_CornerKick_Their,* vykoná sa správanie identifikované ako súperov rohový kop. Ak sa hráč nachádza v hernom móde *PM_CornerKick_Their,* vykoná sa správanie identifikované ako súperov rohový kop. Ak sa hráč nachádza v hernom móde *PM_FreeKick_Their* alebo *PM_IndFreeKick_Their,* vykoná sa správanie identifikované ako súperov voľný kop, súperov individuálny voľný kop. Ak sa hráč nachádza v hernom móde *PM_GoalrKick_Their,* vykoná sa správanie identifikované ako súperova „päťka“. Ak sa hráč nachádza v hernom móde *PM_KickIn_Their,* vykoná sa správanie identifikované ako súperove vhadzovanie. Ak sa hráč nachádza v hernom móde *PM_KickOff_Their,* vykoná sa správanie identifikované ako súperove zahrávanie. Ak sa hráč nachádza v hernom móde *PM_PlayOn,* vykoná sa správanie identifikované ako herný mód.

Pre správania v if-else blokoch, ktoré nie sú ukončené návratom z funkcie, sa ďalej zistí, či daný hráč môže kopnúť do lopty, t.j. či sa nachádzame v „našom“ hernom móde. Ak sa nachádzame v našom móde, tak bežím na určenú pozíciu a koniec. Ak sa nenachádzame v našom hernom móde zistí, či je možné dať gól šmýkačkou. Ak áno, tak spravím šmýkačku. Ak nemôže spraviť

šmýkačku, pokúsi sa zachytiť loptu. Ak sa mu to nepodarilo, zistí, či sa nenachádza v ofsajde a opustí ho. Ak sa však v ofsajde nenachádza, tak každých 100 cyklov sa vracajú hráči do formácie. Medzi tým hráči, ktorí sú ďaleko od lopty sa k nej natočia a zarovnajú hlavu s telom. Ostatní hráči si vypočítajú pozíciu na prijatie lopty. Útočníci sa rozostavia do formácie dáždnik. Ak vzdialenosť od pozície, na ktorú má daný hráč bežať je väčšia ako 2, tak sa upraví, aby sa nedostal do ofsajdu a beží na túto pozíciu. Inak sa pokúsi uvoľniť od súpera a ak sa nemôžem, tak sa otočí na loptu.

Náš rohový kop

Algoritmus:

Určí sa hráč na rohový kop a jeho pozícia a pozícia všetkých ostatných hráčov (napevno). Týmto správanie končí.

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::RunToPosition()`**

Lokalizácia:

Toto správanie bolo nájdené v dokumentácii Loptoši, ktorý toto správanie aj implementoval.

Náš voľný kop, náš individuálny voľný kop

Algoritmus:

Ide len o rozmiestnenie určitých hráčov, zvyšný ostávajú na svojom predchádzajúcom mieste. Zároveň sa predchádza umiestneniu hráčov v ofsajde.

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::RunToPosition()`**

`PlayerTactics::GetOurGoalie()`

`PlayerTactics::ExitOffside()`

`PlayerSkills::GetOffsideLine()`

`PlayerTactics::LPT_PlayFreeKickPosition ()`

Lokalizácia:
Toto správanie bolo posledne zmenené tímom FIITBA.

Naša „päťka“

Algoritmus:
Ide o rozmiestnenie určitých hráčov, zvyšný ostávajú na svojom predchádzajúcom mieste a nastaví sa či ignorujú loptu.

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerTactics::LPT_PlayFreeKickPosition()`**

Lokalizácia:
Toto správanie bolo posledne zmenené tímom FIITBA.

Naše vhadzovanie

Algoritmus:

Ide len o rozmiestnenie určitých hráčov, zvyšný ostávajú na svojom predchádzajúcom mieste a ignorujú loptu. Týmto správanie končí.

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::RunToPosition()`**

Lokalizácia:
Toto správanie bolo vytvorené tímom Loptoši.

Naše zahrávanie

Algoritmus:
Hráč s číslom desať ide na pozíciu, ostatným sa nastaví, či ignorujú loptu.

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::RunToPosition()`**

Lokalizácia:
Toto správanie bolo vytvorené tímom Loptoši.

Náš pokutový kop

Algoritmus:
Nastaví sa, aby hráči ignorovali loptu.

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

Lokalizácia:
Toto správanie bolo vytvorené tímom Loptoši.

Súperov rohový kop

Algoritmus:

Určí sa pozícia všetkých hráčov. Týmto správanie končí.

Závislosti:

objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::RunToPosition()`**

Lokalizácia:
Toto správanie bolo vytvorené tímom FC Farmári.

Súperov voľný kop, Súperov individuálny voľný kop

Algoritmus:

Ide len o rozmiestnenie určitých hráčov, zvyšný ostávajú na svojom predchádzajúcom mieste, nastaví sa či hráči ignorujú loptu. Rozmiestnenie je závislé od pozície lopty.

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::RunToPosition()`**

Lokalizácia:
Toto správanie bolo vytvorené tímom FC Farmári. V dokumentácii nebolo lokalizované, že by niektorý tím toto správanie menil.

Súperova "päťka"

Algoritmus:
Nastaví sa, že hráči neignorujú loptu.

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

Lokalizácia:
V dokumentácii sa správanie nevyskytuje.

Súperove vhadzovanie

Algoritmus:
Ide o rozmiestnenie určitých hráčov, zvyšný ostávajú na svojom predchádzajúcom mieste a nastaví sa ignorovanie lopty. Týmto správanie končí.

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::RunToPosition()`**

Lokalizácia:
V dokumentácii sa správanie nevyskytuje.

Herný mód

Algoritmus:

Zistí sa či je hráč v ofsajde ak áno dostane príkaz ho opustiť a následne koniec. Ak hráč nie je v ofsajde zistí sa kde sa nachádza lopta a podľa toho sa nastaví obrancom ich domovská pozícia. Týmto správanie končí.

Závislosti:
objekty: **`world`** (zistovanie informácií o pozícií, o čísle atď.)

funkcie: **`PlayerSkills::RunToPosition()`**

`PlayerTactics::InOffsidePos()`

`PlayerSkills::GetOffsideLine()`

`PlayerTactics::ExitOffside()`

`PlayerSkills::DependentForBall()` (zisťuje či by mal hráč bežať za loptou)

Lokalizácia:
V dokumentácii sa správanie nevyskytuje.

Zachytenie lopty

Algoritmus:
Zistí sa, či je niekto medzi daným hráčom a loptou, v akom móde sa nachádzame a či je vlastne nutné míňať energiu, ak zahráva náš súper. Vypočíta sa pozícia zachytenia lopty a hráč beží na túto pozíciu. Toto správanie sa vykonáva v metóde *PlayerSkills::InterceptBall().*

Závislosti:
objekty: **`world`** (zisťovanie informácii o móde, lopte a pozícii)

funkcie: **`PlayerSkills::RunToBallInteceptPosition()`**

`PlayerSkills::DependentForBall()` (zisťuje, či by mal hráč bežať za loptou)

`PlayerSkills::GetBallInterceptionPoint()`

Lokalizácia:
Toto správanie vytvoril tím Roztoče.

Utekanie na danú pozíciu

Algoritmus:
Určí sa pozícia a a rýchlosť, na základe rýchlosti koeficient šprintu, overí sa či bod nie je mimo ihriska a natočí sa krk na požadované miesto. Toto správanie sa vykonáva v metóde *PlayerSkills::RunToPosition().*

Závislosti:
objekty: **`world`** (zisťovanie informácii o stamine a pod.)

funkcie: **`PlayerSkills::RecommDashPower()`** (vypočíta doporučený maximálny koeficient šprintu)

`PlayerSkills::TurnNeckTo() (otočenie krku)`

Lokalizácia:
Toto správanie vytvoril tím Roztoče.

Odblokovanie

Algoritmus:
Hráč sa pokúsi presunúť od blokujúceho súpera.

Závislosti:
objekty: **`world`** (zisťovanie informácii o móde, lopte, pozícii a o súperoch)

Lokalizácia:
Toto správanie vytvoril tím Roztoče. Ďalej toto správanie bolo zmenené tímom StjupidDox a nakoniec toto správanie zmenil tím FIITBA.

1.4 Správanie brankára

Loptoši vytvorili nového brankára, pričom v kóde bol zanechaný aj starý brankár. Rozhodli sme sa refaktorovať iba nového brankára vytvoreného tímom Loptošov.

Rozhodovanie sa o činnosti brankára sa realizuje vo funkcii LPT_Main. V ňom sa nachádza niekoľko if blokov. Každý if blok uskutočňuje určité správanie a to buď na základe toho, či je splnená nejaká podmienka, alebo sa priamo v podmienke pokúsi realizovať správanie. Ak sa to podarí, funkcia LPT_Main sa ukončí.

- správanie brankára bez znalosti miesta, kde sa nachádza lopta - `if(!world->ball->IsValid() && world->ball->GetAge() >= 10)`
- vykopávanie lopty po zachytení brankárom alebo pri päťke - `if(world->state->GetPlayMode() == PM_FreeKick_Our || world->state->GetPlayMode() == PM_GoalKick_Our)`
- chytanie lopty - `if(LPT_CatchBall())` - momentálne sa v tomto bloku nevykonáva nič
- kopnutie do lopty - `if(LPT_Kick())` – momentálne sa v tomto bloku nevykonáva nič
- lopta smeruje na bránku alebo je nebezpečná - `if(BallHeadingOnOurGoal() || LPT_DangerousBall())`
- správanie pri prihrávke idúcej v blízkosti bránky - `if(IsPassNearGoal())`
- ak sa nevykoná žiadne z vyššie spomenutých správaní, brankár je v obrannom móde

Správania brankára tímu loptošov boli všetky vytvárané len týmto tímom, preto pri nasledujúcom opise správaní nebude uvádzaná lokalizácia správania v dokumentácii. Namiesto toho bude uvádzaný názov triedy, ktorej funkcia: Behave() predstavuje dané správanie, ktoré sme vytvorili a ktoré je možné nájsť v najaktuálnejšej verzii zdrojových súborov. Naviac v opise uvedieme názov funkcie pôvodne realizujúcej dané správanie. Pripomíname, že všetky správania brankára je sú umiestnené na jednom mieste: ModuleManager/Behaviours/GoalieBehaviours. Rozhodovanie o činnosti brankára (pôvodne realizované vo funkcii LPT_Main) sme umiestnili do najvyššieho správania brankára: PlayGoalieBehaviour. V ňom sa nachádza niekoľko IF blokov. Toto najvyššie správanie podľa vyhodnotenia podmienky v bloku IF volá niektoré z nasledujúcich nami analyzovaných správaní, ktoré budú následne popísané:

1. Správanie bez znalosti o lopte

2. Vykopávanie lopty
3. Správanie pri lopte smerujúcej do brány
4. Správanie pri prihrávke idúcej v blízkosti brány
5. Obranný mód
6. Zrýchlenie v smere tela
7. Chytenie lopty

Správanie bez znalosti o lopte

Názov správania: BallUnknownBehaviour
Funkcia realizujúca správanie: LPT_BallUnknown()

Algoritmus:
Momentálne sa v tomto správaní iba volá nižšie správanie: ScanBehaviour

Závislosti:
objekty: -

funkcie: -

Vykopávanie lopty

Názov správania: KickOffBehaviour
Funkcia realizujúca správanie: LPT_KickOff()

Algoritmus:
Správanie sa vykoná v prípade, že brankár zachytí loptu, prípadne súperov hráč netrafí bránu.

Ak je brankár ďaleko od lopty, tak po ňu zájde (hodnota premennej DATA_STORAGE.goalieKickOff je: GOALIE_GOFORBALL), tento blok CASE sa ukončí nastavením: DATA_STORAGE.goalieKickOff = GOALIE_CAUGHT; (brankár už môže zachytiť loptu). V prípade, že brankár už má loptu, nasleduje otočenie na stred a pozorovanie.

Ak už brankár toho odpozoroval dosť (viac ako 30 cyklov), môžeme aktualizovať svet brankára a ďalej môžeme vykopávať loptu (nastavenie: DATA_STORAGE.goalieKickOff = GOALIE_KICK;). Pri vykopávaní lopty sa len zistí pozícia, kde sa má lopta kopnúť a kopne sa dané miesto. Po vykopnutí lopty sa brankár

dostane do obranného módu (DATA_STORAGE.goalieKickOff = GOALIE_GOHOME), v ktorom je volaná funkcia: LPT_DefenseLine – z nej sme vytvorili správanie DefenceLineBehaviour.

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

`CEvaluatingPlayerList` (zoznam hráčov z vizuálnej informácie hráča)

funkcie: **`int NumberOfEnemiesAround(const Point p,float dist=5.0f)`**

Správanie pri lopte smerujúcej do brány

Názov správania: BallHeadingOnGoalBehaviour
Funkcia realizujúca správanie: LPT_BallHeadingOnGoal()

Algoritmus:
Najprv sa zistí pozícia, z ktorej brankár môže minimálne kopnúť do lopty a na základe toho sa rozhoduje, či je vôbec možné loptu zachytiť (či tá pozícia nie je príliš ďaleko). Ak aj je táto pozícia ďaleko, brankár sa pokúsi zachytiť loptu aspoň na bránkovej čiare. Zachytenie lopty sa realizuje volaním funkcie: LPT_Catch().

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`bool BallHeadingOnOurGoal(float tolerance = 10.f)`**

`bool LPT_DangerousBall()`

`Point GetNearestBallInterceptPos()`

`bool InPenaltyArea(const Point p)`

Správanie pri prihrávke idúcej v blízkosti bránky

Názov správania: `PassNearGoalBehaviour()`

Funkcia realizujúca správanie: LPT_PassNearGoal()

Algoritmus:

Najprv sa zistí pozícia, z ktorej brankár môže minimálne kopnúť do lopty a na základe toho sa rozhoduje, či má brankár utekať na vypočítanú pozíciu (v súčasnosti správanie: RunToPositionBehaviour) alebo beh na zadnú pozíciu čelom vpred – v súčasnosti túto funkcionalitu realizuje metóda:

RunToPositionBehaviour::BehaveBackwards(const Point& target, bool hurry, bool forLife, float positionTolerance, int angleTolerance).

Závislosti:
objekty: **`world`** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **`Point GetNearestBallInterceptPos()`**

`bool IsPassNearGoal()`

Obranný mód

Názov správania: DefenseLineBehaviour()

Funkcia realizujúca správanie: LPT_DefenseLine

Algoritmus:

Ak sa lopta nachádza naboku od brány, brankár sa buď presunie na vhodnú pozíciu, alebo ju len sleduje, ak je na vhodnej pozícii. Ak nie je na vhodnej pozícii, t.j. buď je moc vpredu, alebo moc vzadu, tak sa presunie na pozíciu. Ak na nej už je, tak len presúva bez toho, aby otočil telom.

Závislosti:
objekty: **world** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **bool TurnBody(bool left,float odch);**

void LookAtBall();

void GoalieTurn(float turnMoment);

float GetBestDash(const Point& pointFuture, const Myself& me, Point &bestPoint, float step);

void LPT_Dash(float dashPower);

Zrýchlenie v smere tela

Názov správania: DashBehaviour()

Funkcia realizujúca správanie: LPT_Dash(float dashPower)

Algoritmus:

Vykoná sa príkaz servera: Dash(dashPower) a naviac sa brankár pozrie za loptou.

Závislosti:
objekty: **world** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: **float LPT_GetBestDash(const Point& pointFuture,const Myself& me,Point &bestPoint,float step)**

Chytanie lopty

Funkcia realizujúca správanie: LPT_Catch()

Názov správania: CatchBehaviour()

Algoritmus:

Je to len obalenie elementárneho správania, v ktorom sa najprv na základe pozície brankára vypočíta uhol a zavolá sa príkaz na server:

commandGenerator->Catch(uhol);.

Závislosti:
objekty: **world** (zisťovanie informácií o pozícií, o čísle atď.)

funkcie: -

Príloha C – Návod na vytvorenie a použitie modulov správania

V tejto prílohe uvádzame prevzatú časť z dokumentácie UTTP hráča, za účelom vytvorenia ucelenej a prehľadnej technickej dokumentácie k hráčovi Kukuričné deti, ktorý z hráča UTTP vychádza.

1 Návod na vytvorenie a použitie modulov

1.1 Vytvorenie modulu správania

V priečinku Player/ModuleManager/Behaviours sa nachádzajú vytvorené moduly správania. Nové správanie je tiež potrebné vytvoriť v tomto priečinku. Nové správanie musí dediť od triedy ModuleInterface a prekrývať metódu Behave(), v ktorej bude implementované samotné správanie. Metóda Behave() môže mať viacero parametrov, ale nemá návratovú hodnotu. Tiež je vhodné ošetriť viacnásobné vkladanie hlavičky.

Šablóny pre vytvorenie správaní sa nachádzajú v priečinku Player/ModuleManager/Behaviours/sablona.

1.2 Použitie modulu správania

Do hlavičkového súboru triedy, v ktorej chceme modul použiť, vložíme hlavičkový súbor modulu. V definícii tejto triedy v časti private vložíme makro USE_MODULE(*MenoTriedyModulu*), kde *MenoTriedyModulu*je názov triedy modulu. Toto makro vytvorí metódu Get*MenoTriedyModulu*(). V zdrojových kódoch, kde chceme použiť toto správanie zavoláme metódu Get*MenoTriedyModulu*()->Behave(*params*), kde *params* sú parameter funkcie.

1.3 Vytvorenie testovacieho modulu správania

V priečinku Player/ModuleManager/Behaviours/PlayerBehaviourTests a Player/ModuleManager/Behaviours/GoalieBehaviourTests sa nachádzajú vytvorené testovacie moduly správania. Nové správanie je tiež potrebné vytvoriť v týchto priečinkoch na základe toho pre koho je správanie určené (brankár alebo hráč) . Nové správanie musí dediť od triedy UttpBehaviourTestCase a prekrývať metódy Test(), PrepareBeforeTest() a CleanUpTest(). Metóda PrepareBeforeTest() slúži na predpripravenie testu. Metóda Test() slúži na samotné vykonanie testu a metóda CleanUpTest() sa vykoná po skončení testu. Tiež je možné prekryť metódy OnTestPassed() a OnTestFailed(), ktoré sa vykonajú ak test bol úspešný, resp. neúspešný.

1.4 Použitie testovacieho modulu správania

Podobne ako pri moduloch spravaní sa do hlavičkového súboru triedy ActTestBehaviour vloží hlavičkový súbor testovacieho modulu. V definícii tejto triedy vložíme makro USE_MODULE(*MenoTriedyTestovaciehoModulu*), kde *MenoTriedyTestovaciehoModulu*je názov triedy testovacieho modulu. Toto makro vytvorí metódu Get*MenoTriedyTestovaciehoModulu*(). Volanie testu pridáme do metódy Behave v triede ActTestBehaviour. Test sa zavolá volaním metódy Get*MenoTriedyModulu*()->`RunTest`(*cycles*), kde *cycles* je maximálny počet cyklov počas ktorých bude test bežať. Ak po uplynutí tohto počtu cyklov testovanie neskončilo, tak sa ukončí a začne sa vykonávať nasledujúce správanie. Aby sa všetky testy nevykonávali naraz je potrebné vložiť konštrukciu podobnú nasledujúcej:

```
if(actualTest == 1){

    if(GetMenoTriedyModulu()->RunTest(200)) actualTest++;

}
```

Príloha D – Inštalácia servera 13 pod CygWin

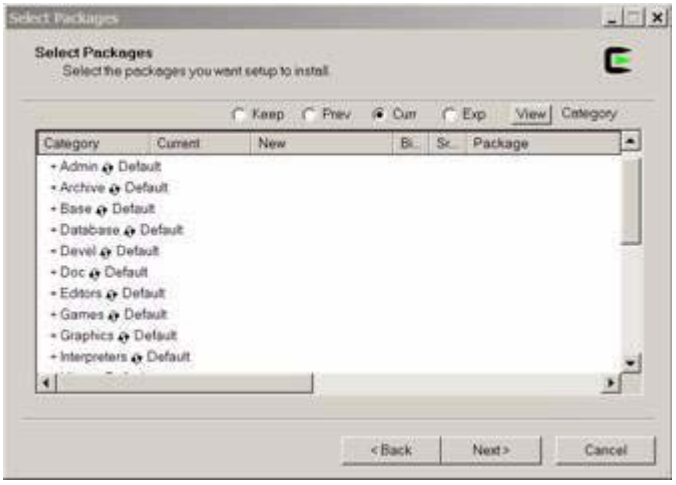
1 Inštalácia servera 13.2.1 a cygwin

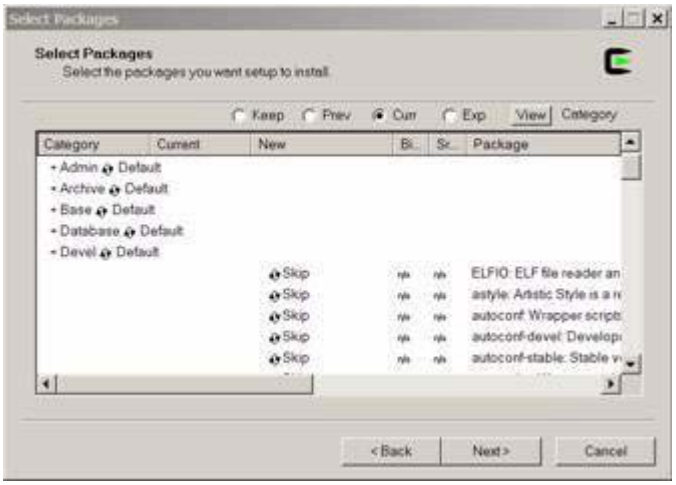
Toto riešenie nezabezpečí, že server nepadne, ale zvyčajne dokáže zahrať celý prvý polčas bez pádu. Toto umožní testovať hráča na servere 13.2.X aj pod windowsom. Výhodné je vypnúť všetky procesy, ktoré nesúvisia s RoboCupom ako winamp, operu a pod. (Hrozí, že server spadne aj skôr ako koniec prvého polčasu). Tento návod bol testovaný pod win XP, ale rovnako by mal fungovať aj na Viste. Konfigurácia pc: Athlon XP 2000+ (1600, taktovaný na 2000) 1.5 GB RAM)

Z tejto stránky si stiahneme najnovší server rozbalíme ho a vložíme najlepšie priamo do C:/sserver, lebo cygwin má problémy s medzerami a názvoch adresárov (napr: Documents and settings)

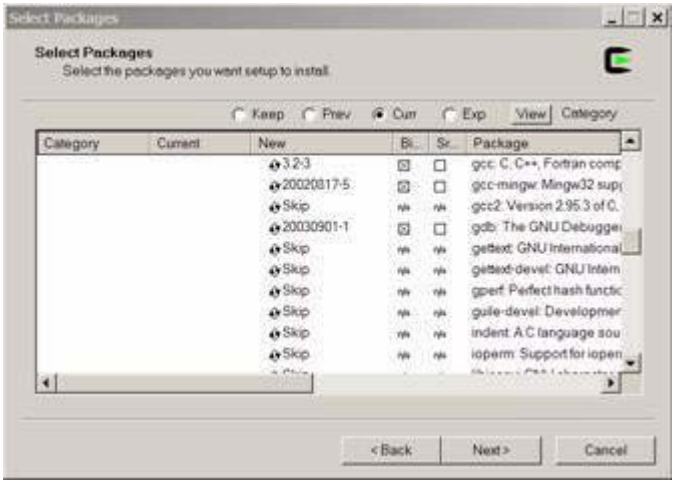
http://sourceforge.net/project/showfiles.php?group_id=24184&package_id=16551

Nainštalujeme cygwin v kolónke devel začiarkneme gcc compiler. Podrobnejší postup tu:





Obrázok 28 . Inštalácia cygwinu



Obrázok 29 Inštalácia cygwinu

http://cygwin.com/setup.exe

Spustíme cygwin a pomocou príkazov cd vojdeme do koreňového adresára C:\rcssserver-13.2.1\rcssserver-13.2.1

Tam spustíme konfiguráciu cez „./configure“

Make „make“

Install „make install“

Keď všetko prebehne bez chýb máme nainštalovaný server. Potencionálne chyby sú

V ceste do serveru máme medzeru, ako bolo už hore písané

Na počítači je nainštalované Logitech zariadenie, to zvyčajne prekáža cygwinu

Iné problémy neboli pozorované

Server po nainštalovaní spúšťame cez cygwin príkazom „rcssserver“ potom už len pripojíme monitor a hráčov normálnym spôsobom.

Príloha E – Vytvorenie liveCD servera 13

1 Vytvorenie liveCD s RoboCup serverom

Zo stránky http://www.slax.org sme si stiahli aktuálnu verziu live distribúcie GNU/Linux Slax, verziu 6.1.1. Táto distribúcia obsahuje nástroje na výrobu modulov pre livecd. Je kompatibilná s distribúciou GNU/Linux Slackware 12.1 , takže je možné použiť balíčky pre ňu určené.

Nabootujeme livecd bežným spôsobom. Na stránke http://packages.slackware.it/ , http://www.linuxpackages.net ,alebo inej stránke obsahujúcej binárne balíčky pre slackware stiahneme balíčky, ktoré sú potrebné pre kompiláciu a spustenie RoboCup servera.:

- flex
- bison
- boost

Tieto balíčky nainštalujeme na ramdisk príkazom install_package, aby kompilačné skripty vedeli nájsť cestu k požadovaným knižniciam napríklad:

#install_package flex-2.5.33-i486-3.tgz

Z týchto balíčkov vytvoríme aj moduly pre livecd príkazom tgz2lzm napríklad:

#tgz2lzm bison-2.3-i486-1.tgz bison-2.3-i486-1.lzm

Tieto moduly si odložíme na médium, na ktorom po reboote ostanú zachované.

Pristúpime ku kompilácii robocup servera, Zo stránky http://sourceforge.net/projects/sserver/ stiahneme aktuálnu verziu zdrojových kódov, v case písania tohto návodu bola dostupná táto verzia zdrojových kódov - rcssserver-13.2.1.tar.gz Archív rozbalíme a skompilujeme príkazmi

#./configure

#make

#make install -Destdir=/tmp/robocup

Vytvoríme balíček

#cd /tmp/robocup

#dir2lzm . robocup.lzm

Balíček si opäť odložíme na médium, ktoré sa nezmaže po reštarte počítača.

Reštartujeme počítač a nabootujeme do svojho oblúbeného operačného systému. Skopírujeme obsah livecd do adresára na disku. Napríklad /tmp/livecd.

Na skopírované livecd, do adresára slax/modules pridáme moduly s príponou .lzm, ktoré sme si odložili.

Po nakopírovaní modulov spustíme skript slax/make_iso.bat alebo slax/make_iso.sh podľa toho, či sa nachádzame v operačnom systéme GNU/Linux, alebo MS Windows.

#./make_iso.sh slax_robocup.iso

Vzniknutý image livecd (slax_robocup.iso) napálime na CD.

Príloha F – Používateľská príručka

1 Používateľská príručka

Táto kapitola obsahuje návod ako spustiť a sledovať simuláciu zápasu dvoch tímov v simulovanom robotickom futbale RoboCup. Základom je mať k dispozícii balíček *Robocup_Simulation_Package.* Tento balíček je súčasťou odovzdaného produktu v rámci predmetu Tvorba softvérového systému v tíme.

1.1 Minimálne požiadavky

Minimálne požiadavky na spustenie simulácie sú:

- Operačný systém *Windows Xp* alebo *Vista*
- Procesor minimálne 500MHz
- 1GB RAM
- 50 MB voľného priestoru na disku

Balíček *Robocup_Simulation_Package* je potrebné nakopírovať priamo na koreňový disk *C:*. Absolútna cesta k tomuto balíčku teda bude *C:\Robocup_Simulation_Package*. V prípade ak nie je možnosť umiestniť tento balíček na disk C:\, je potrebné zmeniť absolútne cesty v jednotlivých *.bat* súboroch.

1.2 Adresárová štruktúra balíčka

Štruktúra balíčka *Robocup_Simulation_Package* obsahuje v základnej podobe nasledujúce komponenty:

- Robocup server (verzia 9.3.6)
- Soccermonitor (monitor pre sledovanie simulácie)
- Hráč Kukuričné deti

- Hráč UTTP

Adresárová štruktúra balíčka je nasledujúca:

./Robocup_Simulation_Package/_monitor/

Obsahuje príslušný monitor, ktorý sa použije pre sledovanie priebehu zápasu. Názov spúšťacieho *.exe* súboru monitora musí byť *monitor.exe*.

./Robocup_Simulation_Package/_server/

Obsahuje príslušný server, ktorý je základom hry (vytvára prostredie hry, udržiava informácie o objektoch v hre, atď.). Názov spúšťacieho *.exe* súboru servera musí byť *rcssserver.exe*

./Robocup_Simulation_Package/play/

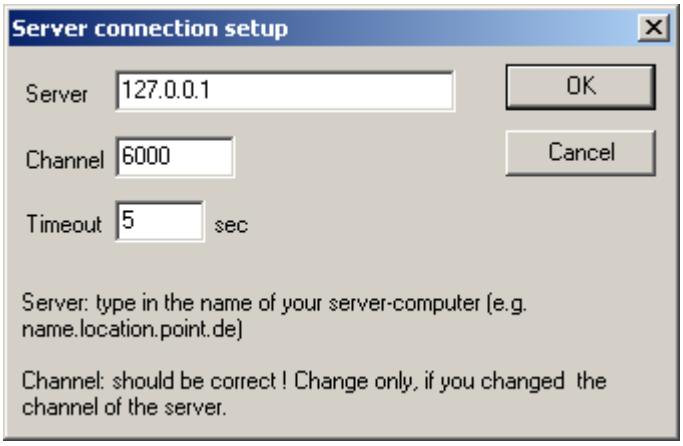
Obsahuje spúšťacie *.bat* súbory, ktoré spúšťajú simulácie zápasov. Každý jeden *.bat* súbor nesie názov podľa toho, aké tímy proti sebe hrajú zápas, pričom vždy tento reťazec začína reťazcom *PLAY_*. Napríklad *PLAY_UTTP_KukuricneDeti.bat* znamená, že proti sebe hrajú tímy *UTTP* a *Kukuričné deti*.

1.3 Spustenie simulácie

Celkové spustenie simulácie s použitím *Robocup_Simulation_Package* sa vykoná spustením akéhokoľvek *.bat* súboru začínajúcim reťazcom *PLAY_*, ktorý je umiestnený v adresári *./Robocup_Simulation_Package/play/*. Jednotlivé názvy sa intuitívne líšia iba tým, ktoré dva tímy proti sebe hrajú.

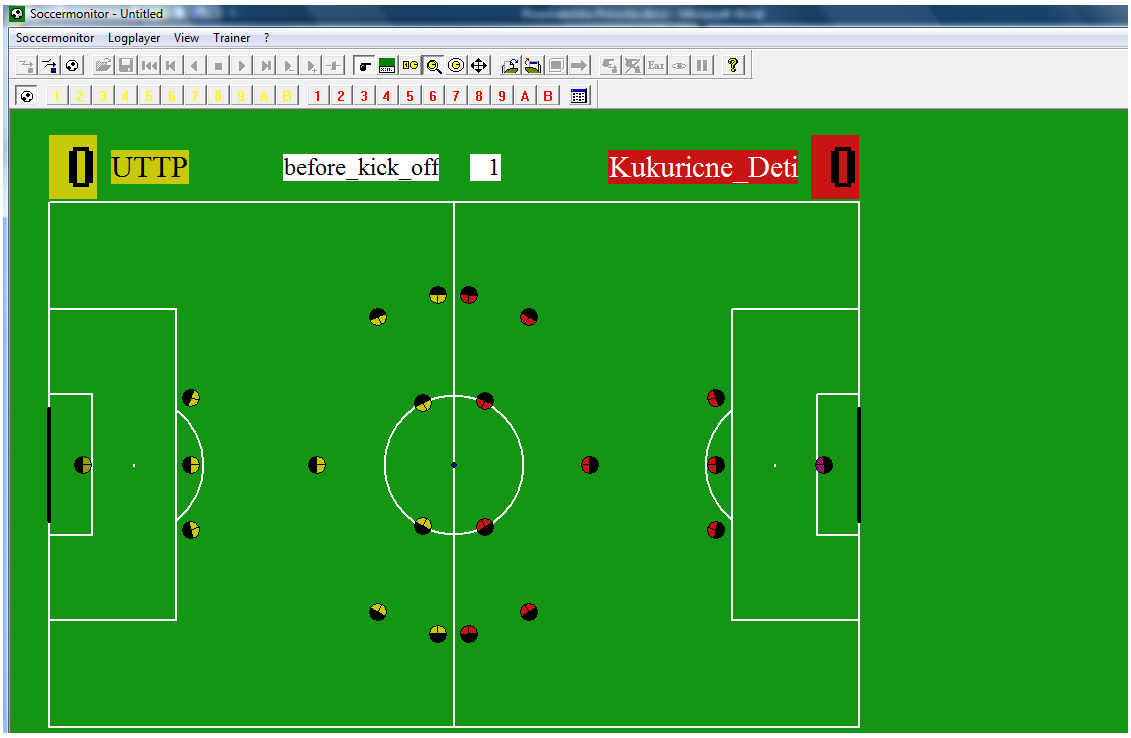
Po spustení príslušného *.bat* súboru sa spustí RoboCup server, RoboCup monitor a dva tímy. Pre spustenie a sledovanie simulácie je nutné pripojiť monitor k serveru a spustiť simuláciu nasledujúcou postupnosťou krokov:

1. Kliknite na ikonu  v ľavej hornej časti monitora tak, ako ju je vidieť na obrázku 31. Otvorí sa nám dialógové okno, ktorého parametre treba vyplniť tak, ako je to vidieť na obrázku 30.



Obrázok 30 Dialógové okno pre pripojenie monitora k serveru

2. Treba počkať kým sa nezobrazia na hracej ploche všetci 11 hráči na oboch stranách, ak sa tak po pripojení monitora ešte nestalo. Príklad, keď sú už všetci hráči načítané v hre, ukazuje obrázok 31. Pre spustenie simulácie je potrebné kliknúť na ikonu  , nachádzajúcu sa v ľavom hornom roku monitora.



Obrázok 31 Situácia pred spustením simulácie zápasu – všetci hráči sú načítaní a pripravení hrať.

Sledovanie a ovládanie simulácie

Po spustení oboch tímov, by sa ich mená mali zobraziť v monitore ako je to na obrázku A-2. Povoliť výkop zo stredu ihriska kliknutím na ikonu  je potrebné po každom góle a po polčase. Nástroj *Soccermonitor* (monitor pre sledovanie simulácie) obsahuje aj ďalšie užitočné funkcie, ktoré umožňujú modifikovať simuláciu podľa požiadaviek ale na spustenie simulácie nie sú nevyhnutné. Tento nástroj ďalej umožňuje napríklad uložiť si záznam zápasu a neskôr si ho opätovne prehrať.

1.4 Štatistika zápasu

Kouč tímu vytvára štatistiku z každého zápasu, ktorá sa nachádza v adresári *Debug*. Ak tento adresár neexistuje, je ho potrebné manuálne vytvoriť. V tomto adresári je jeden súbor pre každého hráča zápasu s podrobnou štatistikou a aj súbor *Timova statistika.txt,* v ktorom sú zhrnuté základné údaje o každom hráčovi v zápase spolu so súhrnnými štatistikami oboch tímov. Štatistiky vytvára aj release, aj debug verzia kouča.

Štatistika hráčov sa vytvorí vždy, keď je korektne ukončené spustenie kouča. Kouč ukončí svoju činnosť po vypnutí simulačného servera. Takýmto spôsobom je možné získavať štatistiky za zápas ukončený v riadnom hracom čase, ale aj za zápas, ktorý bol predčasne ukončený.

1.5 Spustenie liveCD s RoboCup serverom

LiveCD s RoboCup serverom zasunieme do CD-ROM mechaniky, uistíme sa, či máme v BIOS-e povolené bootovanie z CD. Po načítaní liveCD nabehne bootovacie menu, kde si môžeme zvoliť grafické prostredie, alebo textovú konzolu. Pre úsporu prostriedkov počítača odporúčam spustiť textovú verziu.

Grafická verzia prihlási používateľa automaticky, v textovej konzole je nutné zadať používateľa “root” a heslo “toor”.

Ak je v sieti spustený DHCP server, IP adresa počítača je nakonfigurovaná automaticky. Inak treba nakonfigurovať IP adresu príkazom ifconfig.

#ifconfig <IP adresa> netmask <,maska siete>

Ak funguje sieť, spustíme server príkazom

#rcssserver

Upozornenie – LiveCD beží pod používateľom root, preto si dávajte pozor na bezpečnosť

Príloha G – Vizualizácia rozhodovacích stromov správania sa hráča

1 Strom pre herne módy

Na nasledujúcich obrázkoch vidno správanie sa hráča, ktoré je rozdelené na správanie, keď má súper loptu a keď má loptu náš hráč. Tieto správania sa vykonávajú, keď nastane špecifický herný mód ako rozohrávanie, aut atď. Tento strom nebol zmenený. Zmeny sa zavádzali len v rámci odstránenia slepého návratu našich hráčov na pozície hlboko v našej obrane, hoci sme boli v útoku a žiadna hrozba zo strany súpera nebola prítomná.

2 Strom správania sa hráča UTTP

Na nasledujúcich obrázkoch vidno správanie sa hráča, ktoré vykonáva, keď má loptu a hra pokračuje ďalej bez prerušenia. Pôvodný strom obsahuje rôzne nedostatky dokonca aj hrubé chyby v základnej hráčskej stratégii. Za hrubú chybu možno označiť snahu o prihrávku ešte predtým ako by sa najprv hráč pokúsil loptu posunúť viac dopredu. Takáto chyba je bežná dokonca aj pri reálnej hre (zvyčajne malých detí), kedy sa slabší hráči snažia zbaviť lopty vzápätí ako ju dostanú, aby loptu nestratili.

3 Strom správania sa hráča Kukuričné deti

Na nasledujúcich obrázkoch vidno upravené správanie hráča, ktoré hráč vykonáva, keď má loptu a hra pokračuje bez prerušenia. Vyššie spomenuté chyby boli odstránené, teda hráč sa pokúsi najprv loptu posunúť dopredu driblovaním a až keď nemá miesto, tak sa snaží o prihrávku svojmu spoluhráčovi.

Príloha H – Nahradené časti dokumentácie v letnom semestri

Táto kapitola obsahuje tie časti dokumentácie zo zimného semestra 2008/2009, ktoré sme boli nútení odstrániť z dôvodu ich aktualizácie. Jedná sa o nasledujúce kapitoly inžinierskeho diela: *Obsah*, *Úvod*, *Zhodnotenie a Použitá literatúra*.