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

RoboCup S - Nové stratégie

Prototyp

verzia 0.1

Obsah dokumentu

1 Úvod

V tomto dokumente popisujeme prototyp hráča tímu Jahodový princovia. Podkladom pre vytváranie novej verzie hráča sa stal existujúci a dostupný kód od tímu Gang of Six. V rámci časových možnosti boli do tohto prototypu implementované niektoré časti, ktoré boli popisované v časti hrubý návrh. Jednotlivé časti, ktorým sme sa venovali v prototype hráča sú popísane v nasledujúcej časti dokumentu.

1.1 Prehľad dokumentu

V nasledujúcej kapitole je opísaná zmena vo funkčnosti brankára. Nachádza sa tam stručný popis implementácie, ktorá bola vykonaná v rámci prototypu, a popis časti neimpelentovanej z časového hľadiska. Táto bude doplnená vo finálnej verzií.

V kapitole číslo tri je rozobratá problematika kouča. Jeho samotná implementácia a možnosti, ktoré so sebou daná implementácia prináša.

Kapitola číslo štyri rieši problematiku heterogénnych hráčov, ktorí sú súčasťou predkladaného prototypu hráča.

Samotný záver tejto časti dokumentu pojednáva o implementácii a vylepšení algoritmu na prihrávanie medzi spoluhráčmi nášho prototypu.

1.2 Status rozpracovanie úloh

Nasledujúca tabuľka poukazuje na aktuálny stav práce na jednotlivých kapitolách týkajúcich sa nášho hráča.

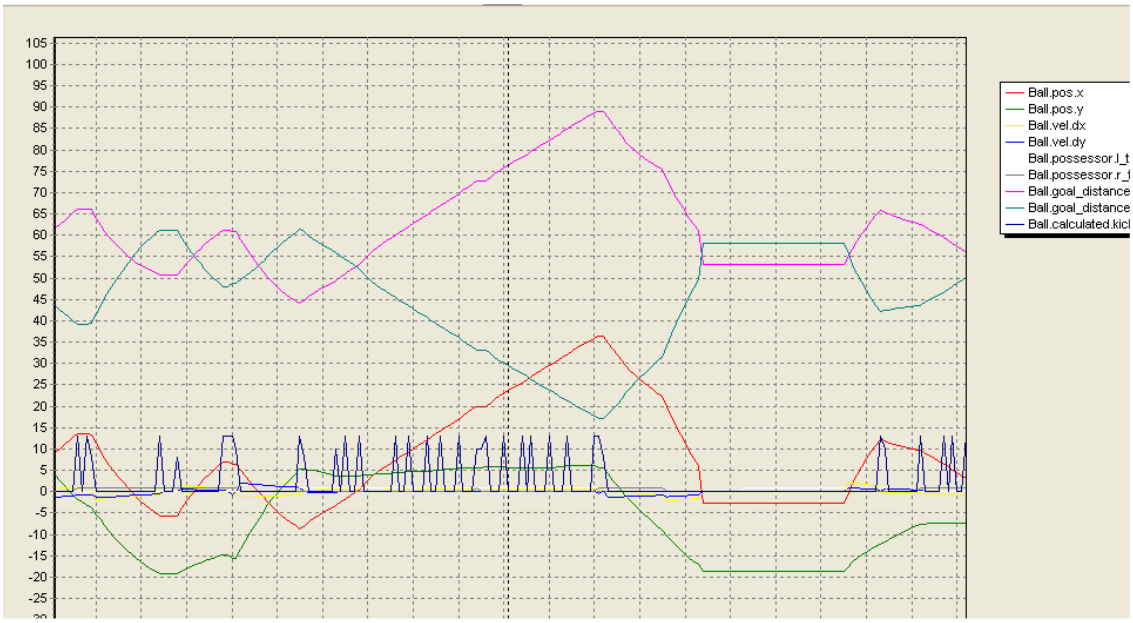
Tab. 1 Status rozpracovania úloh

Úloha	**Stav plnenia**
Nastavenie a spustenie heterogénnych hráčov	implementácia je ukončená, plne zodpovedá návrhu a plne funkčná.
Striedanie heterogénnych hráčov počas zápasu	problém je dobre zanalyzovaný a riešenie navrhnuté, implementácia je naplánovaná do ďalšieho semestra.
Zmena konštanty výbehu brankára	konštanta zmenená, vykonané testy po zmene
Zmena spôsobu návratu brankára do brány	prepracovaná analýza návrhu, väčšia časť bude implementovaná v 2. semestri
Nahrávky od brankára	vylepšenie a spresnenie nahrávok
Nahrávky od brankára	Zhustenie lúčov pred brankárom je implementované a otestované.
Kouč	Implementovaná kompletná funkčnosť

2 Logalyzer – nástroj na prezeranie logovacích súborov

Aby sme vedeli odhaliť nedostatky už implementovaného riešenie tímu Gang of Six, tak sme sa rozhodli preskúmať dopodrobna riešenie, ktoré sme prevzali. Aj kvôli tomu sme sa rozhodli pracovať s nástrojom, ktorý nám umožňuje získať podrobné a veľmi užitočné informácie, ktoré môžeme využiť v ďalšej našej práci. V nasledujúcich riadkoch opíšme dôvod výberu a vôbec použitia takéhoto nástroja ako aj jeho využitie.

Po každom odohranom zápase vytvorí server záznam o zápase – tzv. logovací súbor. Logovacie súbory sú dôležitým prostriedkom sledovania priebehu zápasu a vyhodnocovania údajov zozbieraných počas zápasu. Samotne logovacie súbory nieje možne čítať klasickými textovými editormi. Preto je potrebne nájsť nástroj ktorý dokáže vyhodnocovať tieto logovacie súbory. Po vyskúšaní viacerých nástrojov na prezeranie logovacích súborov sa tím rozhodol používať nástroj Logalzyer. Hlavnými dôvodmi pre výber tohto nástroja boli prehľadne užívateľské prostredie, jednoduché ovládanie a množstvo užitočných funkcií. V ďalšej časti kapitoly sú opísane najpoužívanejšie funkcie nástroja Logalyzer.

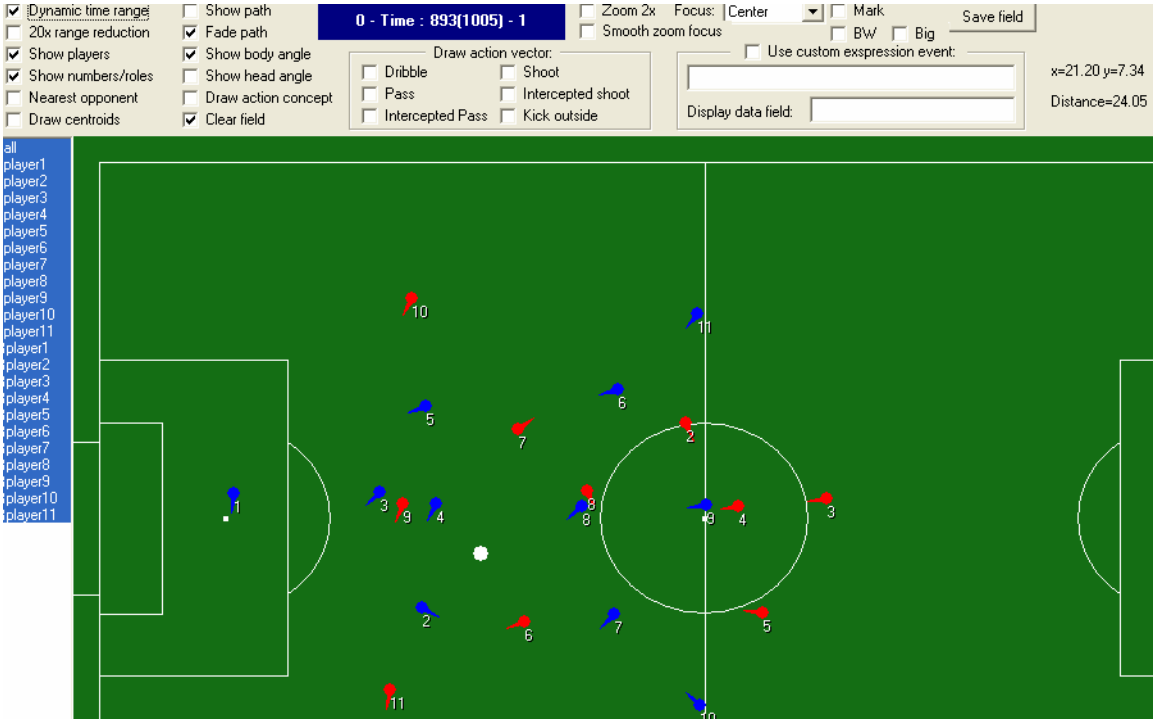


Obr. 1 Sledovanie grafov pomocou nástroja Logalyzer

V záložke Chart v hlavnom menu nástroja sa nachádza okno umožňujúce sledovať grafy reprezentujúce zvolene veličiny. Záložka Chart je zobrazená na Obr. 1. Zvoliť na sledovanie je možne akúkoľvek veličinu pohyblivých objektov (rýchlosť, sila streľby, výdrž). Okrem jednotlivých veličín udávajúcich stav hráčov na ihrisku nástroj umožňuje sledovať aj stav skóre, aktuálnu pozíciu lopty či čas. Veličiny, ktoré budú sledovane je možne ľubovoľne kombinovať.

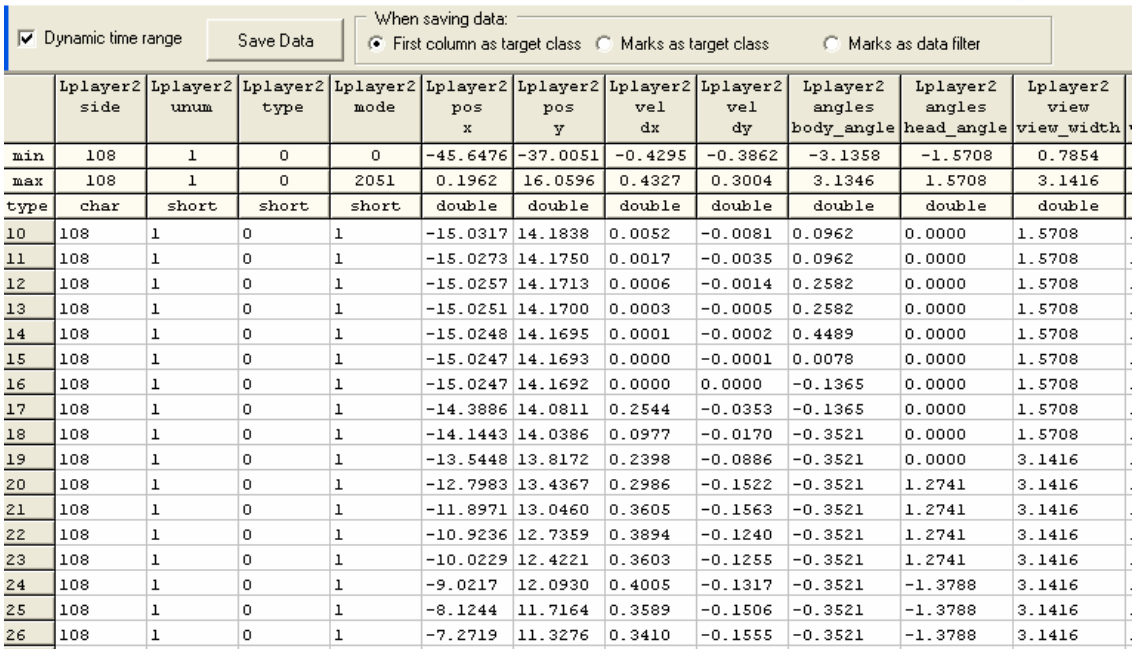
Veľmi často využívanou funkciou je aj sledovanie zápasu prostredníctvom tzv. okna s ihriskom („field window“). Táto funkcia nástroja Logalyzer je zobrazená na Obr. 2. Okno

s ihriskom dokonale simuluje rôzne monitory zápasov, ktoré sa používajú na sledovanie zápasov priamo počas toho ako sa odohrávajú. Tieto monitory sa však už nedajú použiť na sledovanie zápasov z logovacích súborov. Nástroj Logalyzer takéto sledovanie zápasov umožňuje a dokonca ponúka množstvo atribútov, ktorými je možne upraviť si pohľad na ihrisko (napr. zobrazovať čísla hráčov, zobrazovať uhol tela, zobrazovať uhol pohľadu...)



Obr. 2 Monitor zápasov

Poslednou z najpoužívanejších funkcií nástroja je tzv. analyzátor dát. Tento analyzátor je možne spustiť zo záložky Data. Analyzátor dát zobrazuje dáta v tabuľkovej forme. Dáta, ktoré sa budú sledovať je rovnako ako v prípade grafov možne ľubovoľné kombinovať. Analyzátor dát je zobrazený na Obr č. 3.



Dynamic time range | Save Data | When saving data: First column as target class | Marks as target class | Marks as data filter

	Lplayer2 side	Lplayer2 unum	Lplayer2 type	Lplayer2 mode	Lplayer2 pos x	Lplayer2 pos y	Lplayer2 vel dx	Lplayer2 vel dy	Lplayer2 angles body_angle	Lplayer2 angles head_angle	Lplayer2 view view_width
min	108	1	0	0	-45.6476	-37.0051	-0.4295	-0.3862	-3.1358	-1.5708	0.7854
max	108	1	0	2051	0.1962	16.0596	0.4327	0.3004	3.1346	1.5708	3.1416
type	char	short	short	short	double	double	double	double	double	double	double
10	108	1	0	1	-15.0317	14.1838	0.0052	-0.0081	0.0962	0.0000	1.5708
11	108	1	0	1	-15.0273	14.1750	0.0017	-0.0035	0.0962	0.0000	1.5708
12	108	1	0	1	-15.0257	14.1713	0.0006	-0.0014	0.2582	0.0000	1.5708
13	108	1	0	1	-15.0251	14.1700	0.0003	-0.0005	0.2582	0.0000	1.5708
14	108	1	0	1	-15.0248	14.1695	0.0001	-0.0002	0.4489	0.0000	1.5708
15	108	1	0	1	-15.0247	14.1693	0.0000	-0.0001	0.0078	0.0000	1.5708
16	108	1	0	1	-15.0247	14.1692	0.0000	0.0000	-0.1365	0.0000	1.5708
17	108	1	0	1	-14.3886	14.0811	0.2544	-0.0353	-0.1365	0.0000	1.5708
18	108	1	0	1	-14.1443	14.0386	0.0977	-0.0170	-0.3521	0.0000	1.5708
19	108	1	0	1	-13.5448	13.8172	0.2398	-0.0886	-0.3521	0.0000	3.1416
20	108	1	0	1	-12.7983	13.4367	0.2986	-0.1522	-0.3521	1.2741	3.1416
21	108	1	0	1	-11.8971	13.0460	0.3605	-0.1563	-0.3521	1.2741	3.1416
22	108	1	0	1	-10.9236	12.7359	0.3894	-0.1240	-0.3521	1.2741	3.1416
23	108	1	0	1	-10.0229	12.4221	0.3603	-0.1255	-0.3521	1.2741	3.1416
24	108	1	0	1	-9.0217	12.0930	0.4005	-0.1317	-0.3521	-1.3788	3.1416
25	108	1	0	1	-8.1244	11.7164	0.3589	-0.1506	-0.3521	-1.3788	3.1416
26	108	1	0	1	-7.2719	11.3276	0.3410	-0.1555	-0.3521	-1.3788	3.1416

Obr. 3 Analyzátor dát

Nástroj Logalyzer sa už počas vývoja prototypu ukázal ako veľmi užitočný a tím vyslovil s týmto nástrojom spokojnosť.

3 Heterogénni hráči

Kapitola *heterogénni hráči* bola popísaná v hrubom návrhu, ktorý bol súčasťou projektovej dokumentácie za zimný semester. V procese implementácie logicky nasledovala za krokom vytvorenia, konfigurovania a spustenie kouča a tried, ktoré s ním priamo súvisia, ktoré ho implementujú.

Ako bolo v tejto kapitole okrem iného opísané, na začiatku futbalového zápasu má kouč možnosť vybrať si zo siedmych typov hráčov, ktorí majú rôzne vlastnosti. Tieto typy hráčov značia, ktoré vlastnosti hráča sú vylepšené a naopak, ktoré zase degradované. Každý heterogénny hráč má niektorú schopnosť na vyššej úrovni (napr. výdrž, zrýchlenie, presnosť nahrávky, sila kopu a podobne) a zároveň niektorú potlačenú. Štandardným typom hráča, ktorý má všetky vlastnosti na jednej úrovni, je hráč typu 0. Tento druh hráča je využívaný tímom Gang of Six, a my pokračujeme v jeho vylepšovaní. Ďalšími možnosťami výberu sú typy od jedna až po typ šesť vrátane. Typy s vylepšenými vlastnosťami sa používajú pri špecifických rolách v tíme.

Tým pádom, že hráč Gang of Six neobsahoval kouča, neexistovala ani možnosť vybrania heterogénnych hráčov na začiatku futbalového zápasu.

Nasledujúca tabuľka ukazuje požadované vlastnosti, ktoré by mali mať jednotlivé roly hráča.

Tab. 2 Pridelenie vlastnosti hráčskym roliam

Rola hráča	Požadované vlastnosti
Obranca	Má za úlohu zastaviť prenikajúceho hráča, a preto by mal vedieť vyvinúť veľkú rýchlosť, a musí loptu odkopnúť preč od vlastnej brány, preto musí vedieť kopnúť loptu veľkou silou, avšak nepotrebuje vedieť loptu odkopnúť presne.
Stredopoliar	Prijíma loptu z obrany a rozohráva ju útočníkom, preto musí vedieť presne prihrávať a mať veľmi kvalitné informácie o aktuálnom svete. Nemusí byť rýchly (mať veľkú akceleráciu) a ani obratný. Mal by však mať veľkú výdrž, keďže cez stredopoliara sa prelína celá hra, od obrany k útoku.
Stredný útočník	Prijíma loptu od stredopoliarov, musí sa s ňou dostať ku bránke a kopnúť gól, preto by mal vedieť vyvinúť väčšiu rýchlosť (akcelerácia) a mal mať veľkú silu kopu. Takisto musí byť obratný, aby ho obrancovia ľahko neobrali o loptu. Tieto vlastnosti môžu byť zvýraznené na úkor presnosti prihrávky a takisto výdrže.
Útočiaci krídelník	Prijíma loptu od stredopoliarov a prihráva ju ďalšiemu útočníkovi, preto musí mať veľkú akceleráciu a spomalenie (na presné zachytenie prihrávok).

3.1 Vyhľadanie typov

Nám sa podarilo implementovať metódy, ktoré nájdu a správne identifikujú heterogénnych hráčov po spustení servera. Tento proces je podstatný, pretože server po každom svojom spustení vygeneruje 6 druhov heterogénnych hráčov náhodne. Ako už bolo spomenuté, každý heterogénny hráč má svoje unikátne vlastnosti, ktoré z technického pohľadu predstavujú hodnoty definované v konfiguračných súboroch (*server.conf* a *player.conf*). Metódy, ktoré tieto vlastnosti vedia rozpoznať, boli implementované v hlavnej triede kouča – *BasicCoach.cpp.*

V prvom rade je potrebné zistiť, či futbalový zápas ešte na začiatku. To sa deje prostredníctvom podmienky:

```
if(WM->getTimeLastSeeGlobalMessage().getTime() == 0)
```

,kde *WM* je inštancia je inštancia triedy *WorldModel*, ktorý implementuje aktuálny svet .

Pre istotu sa tiež overuje, či heterogénni hráči už vystriedaní neboli, pomocou podmienky

```
if(bSubstituted == false)
```

,kde premenná bSubstitued je typu *boolean*.

Nasleduje blok troch cyklov *for*, v ktorých prebieha samotná identifikácia jednotlivých typov heterogénnych hráčov. Príkazy v týchto cykloch majú v podstate podobný charakter, menia sa len jednotlivé hodnoty a parametre.

Tab. 3 Určenie vlastností hráča na základe vygenerovaných parametrov

Parameter hráča	Vlastnosť hráča	Rozsah
Player_speed_max	Rýchlosť	1.0-1.2
Player_decay	Rýchlosť	0.4-0.6
Kickable_margin	Sila kopu	0.7-0.9
Dash_power_rate	Zrýchlenie	0.006-0.008
Effort_min	Zrýchlenie	0.4-0.6
Effort_max	Zrýchlenie	0.8-1.0
Extra_stamina	Výdrž	0-100
Stamina_inc_max	Zotavovanie	25-45
Kick_rand	Presnosť	0.0-0.1
Inertia_moment	Obratnosť	5.0-10.0

3.1.1 Hľadanie stredných útočníkov

V cykle *for* iterujeme od 0 po maximálny počet typov heterogénnych hráčo (7), ktoré náhodne vygeneroval server pri spustení. Následne z *WorldModelu* získame informácie pre každý iterovaný typ príkazom:

```
m_player_types[i] = WM->getInfoHeteroPlayer( i );
```

,kde *m_player_types* je inštanciou triedy `HeteroPlayerSettings`, ktorá je súčasťou triedy *ServerSetting*.

Keďže pre útočníka je dôležitá aktuálna rýchlosť (`calcRealSpeedMax`), obratnosť

(`dInertiaMoment`) a silu kopu (`dKickableMargin`) do premennej *`realSpeed`* si uložíme súčin týchto atribútov:

```
realSpeed = curRealSpeedMax * m_player_types[i].dInertiaMoment * m_player_types[i].dKickableMargin;
```

Metóda `calcRealSpeedMax` s parametrom integer má za úlohu túto hodnotu vyrátať. Najprv sa vyrátat tzv. *effective dash power*, teda hodnota efektévneho pohybu hráča. Daný vzorec je súčasťou manuálu k serveru pre RoboCup jednoducho sme ho iba aplikovali do nášho frameworku a hráča. Je to obyčajný súčin zrýchlenia (`dDashPowerRate`) a maximálnej snahy hráča (`dEffortMax`)

```
float edp = 100.0f * player_type.dDashPowerRate * player_type.dEffortMax;
```

Následne vyrátame reálnu rýchlosť, ktorú dostaneme ako podiel *effective dash power* `(edp)`a zvyškovej rýchlosti hráča `(1.0f - player_type.dPlayerDecay)`.

```
float real_speed_max = edp / (1.0f - player_type.dPlayerDecay);
```

Ďalej už len postupujeme štandardným algoritmom hľadania maximálnej hodnoty z určitej množiny hodnôt. Až takúto hodnotu nájdeme, do premennej s názvom `idForward` si uložíme číslo tohoto typu s maximálnou rýchlosťou.

V prípade, že nájdeme typy s rovnakou maximálnou rýchlosťou, budeme brať potom do úvahy výšku zotavovania – `dStaminaIncMax`.

3.1.2 Hľadanie krídelných útočníkov

Pri krídelných útočníkoch berieme do úvahy okrem ich momentálnej rýchlosti a akcelerácie (`curRealSpeedMax`) aj presnosť prihrávky (`dKickRand`) strednému útočníkovi.

```
realSpeed = curRealSpeedMax * m_player_types[i].dKickRand;
```

Obdobne ako v predchádzajúcom prípade, aj tu budeme hľadať ten správny heterogénny typ pomocou hľadania maximálnej hodnoty `realSpeed`, ktorú potom priradíme premennej `idWing`.

V prípade, že sa nájdu heterogénni hráči s rovnakou výškou realSpeed, vyberie sa ten, ktorý má vyššiu hodnotu výdrže (`dStaminaIncMax`) .

3.1.3 Hľadanie stredopoliarov

Ako vyplýva z Tab. 2, pre stredopoliara je dôležité, aby vedel presne nahrať (`dKickRand`), aby mal vysokú výdrž (`dExtraStamina`) a takisto zotavovanie (`dStaminaIncMax`). Preto tu namiesto maximálnej rýchlosti budeme rátať súčin práve týchto atribútov:

```
curMiddfielderIndex = m_player_types[i].dStaminaIncMax * m_player_types[i].dKickRand * m_player_types[i].dExtraStamina;
```

Ďalej už postupujeme rovnakým algoritmom nájdenia typu s maximálnou hodnotou tohto indexu, ktorú priradíme do premennej `idMiddfielder`.

3.1.4 Hľadanie obrancov

Podobne ako pri procese hľadania stredopoliarov či útočníkov sa pri hľadaní heterogénnych typov obrancov využíva obdobný postup.

Pri obrancoch je treba brať do úvahy najmä rýchlosť a akceleráciu (`curRealSpeedMax`) a rozsah chytenia lopty (`dKickableMargin`).

```
curDefIndex = curRealSpeedMax * m_player_types[i].dKickableMargin;
```

Rovnako ako pri predchádzajúcich postupoch, aj tu sa nájde najvyšší curDefIndex všetkých heterogénnych typov, ktorý sa uložé do premenne s názvom `idDefense`.

3.2 Pridelenie nájdených heterogénnych typov jednotlivým hráčom

V tomto kroku nasleduje samotné vystriedanie hráčov, ktoré prebieha prostredníctvom vykonania príkazu na server.

change_player_type(<unum> <player_type>)

<unum> - Číslo hráča v tíme (obvykle od 1 po 11)

<player_type> - Typ heterogénneho hráča (od 1 po 7)

V našej konkrétnej implementácii to bude vyzerať nasledovne:

Vystriedanie obrancov

```
substitutePlayer(2, idDefense);
substitutePlayer(3, idDefense);
substitutePlayer(4, idDefense);
```

Vystriedanie stredpoliarov

```
substitutePlayer(5, idMiddfielder);
```

Vystriedanie stredných útočníkov

```
substitutePlayer(8, idForward);
```

Vystriedanie útočiacich krídelníkov

```
substitutePlayer(7, idWing);
```

Metóda `substitutePlayer` len implementuje spomínaný príkaz na server pre vystriedanie hráčov *change_player_type*(<unum> <player_type>).

Zoznam hráčov typov hráčov je uložený v súbore *formations.conf*.

4 Kouč

Táto kapitola sa zaoberá implementáciu kouča v prototype projektu.

4.1 Motivácia

Používanie kouča môže priniesť do zápasu heterotypy a tým aj zmenu taktiky hry a vlastnosti jednotlivých hráčov. Našou snahou pri implementovaní kouča a tým aj heterotypov je vylepšiť výsledky podávané našim tímom.

4.2 Implementácia kouča

Podarilo sa nám implementovať kouča, na ktorom pracovali tímy Gang of Six a UvaTrilearn. Zo zdrojových súborov sme zistili, že kouč mohol poskytovať iba jednoduchú metódu na nasadenie heterotypov do hry, avšak bol to iba pokus, ako by sa daná črta mala implementovať do kouča. Nám sa s koučom podarilo:

- Rozbehanie kouča.
- Nasadenie správnych heterotypov pomocou kouča.

4.2.1 Rozbehanie kouča

Rozbehanie kouča si vyžadovalo:

- Nájsť jeho implementáciu v zdrojových súboroch.
- Zistenie funkcionality zdrojových súborov pomocou ich obhliadky.
- Upravenie skriptu pre kompiláciu a linkovanie zdrojových súborov.
- Upravenie a nastavenie skriptov pre spúšťanie zápasov.
- Testovanie.

4.2.1.1 BasicCoach

Základnou triedou, ktorá poskytuje funkcionalitu kouča je trieda BasicCoach (súbor BasicCoach.cpp). V nej sme museli upraviť a doplniť jednotlivé časti kódu tak aby kouč vedel:

- Nájsť z heterotypov dodaných serverom správny heterotyp pre obrancov.
- Nájsť vhodný heterotyp pre obrancu.
- Nájsť vhodný heterotyp pre stredných útočníkov.
- Nájsť vhodný heterotyp pre útočiacich krídelníkov.
- Nasadiť vyššie spomenuté heterotypy do hry.
- Vyrátať maximálnu reálnu rýchlosť hráča.

4.2.1.2 mainCoach

Ďalšou triedou, ktorá ma podiel na činnosti kouča je trieda mainCoach (súbor mainCoach.cpp). Táto trieda spúšťa činnosť kouča. V triede sme museli upraviť komunikáciu kouča so serverom a taktiež jeho vnútorný svet.

4.2.1.3 Buildovací skript

Úprava buildovacie skriptu (súbor makefile) spočívala v správnom nadefinovaní tried, vytvorení kompilovacej jednotky pre kouča a nalinkovanie jednotlivých tried. Výsledkom našich zmien je build súbor, ktorý dokáže skompilovať, zlinkovať a vytvoriť výstupný spustiteľný súbor. Tento súbor slúži po spustení ako kouč.

4.2.1.4 Spúšťanie kouča

Na spúšťanie kouča bolo potrebné vytvoriť skript, ktorý danému *.exe súboru poskytne správne parametre tak, aby kouč vedel:

- Za aký tím hrá.
- Host, kde sa nachádza server.

Ostatné parametre potrebné pre činnosť kouča sú v konfiguračných súboroch alebo priamo v zdrojových kódoch.

4.3 Nasadenie správnych heterotypov

Kouč na začiatku zápasu získa od servera informácie o charakteristikách jednotlivých hráčov. Podľa týchto charakteristík určí jednotlivé heterotypy. Potom tieto správne určené heterotypy nasadí na ihrisko na jednotlivé pozície. Napr. na pozíciu útočníka sa vyžaduje iný heterotyp ako na pozíciu obrancu. Celá táto činnosť sa vykoná pred začiatkom zápasu.

5 Brankár

5.1 Brankár a jeho prihrávky

5.1.1 Nenahrávať obsadeným hráčom

Po analýze algoritmu, ktorý tím Gang Of Six naimplementoval, sme si uvedomili, že má niekoľko nedostatkov. Často sa stávalo, že brankár pri rozohrávke nahral protihráčovi. Určili sme niekoľko dôvodov, prečo k takýmto situáciam dochádza.

Jedným z problémov je, že pre každý lúč sa vyrátava fitness pre najlepšie postaveného spoluhráča a ďalší fitness pre najlepšie postaveného protihráča. Následne tieto dva hodnoty odčíta a výsledná hodnota predstavuje fitness celého lúča. V prípade, že všetci spoluhráči boli krytí protihráčmi, vybral si brankár jedného spoluhráča, ktorý síce mal najlepší fitness, ale v jeho blízkosti bol jeden alebo viac protihráčov. Treba si uvedomiť, že hráči nemajú kompletné informácie o dianí na ihrisku. Môže sa stať, že pri výpočte sú použité už staré informácie, a že sa poloha spoluhráča medzičasom zmenila. Takisto prihrávky (a strely vo všeobecnosti) nemusia byť presné tak, ako si to hráč zvolí, keďže server aj tu vkladá šum. Keď zoberieme do úvahy všetky tieto aspekty, je zrejmé, prečo v niektorých prípadoch brankár nahrá priamo súperovi.

Riešenie tohto problému nie je jednoduché a zdá sa, že sa k ideálnym výsledkom môžme len priblížiť.

V kóde sa často objavuje parameter *dConfThr*, čo je tzv. confidence threshold. Tento parameter sa používa pri iterácii nad množinou objektov (hráčov) vo modeli sveta (trieda WorldModel). Určuje pravdepodobnosť, s akou sa má hráč zahrnúť do množiny spoluhráčov alebo protihráčov. Ako som spomínal, informácie o hráčoch starnú v každom cykle, ak nie sú obnovované. Tento parameter určuje, ktoré informácie (a ktorých hráčov) má zahrnúť do výpočtu. Tím Gang Of Six nastavil pri prihrávkach tento parameter na 0,9. Zmenili sme túto hodnotu pri prihrávkach brankára na vyššiu. Týmto sme zabezpečili, že hráč nahráva len tým hráčom, o ktorých má najčerstvejšie infomácie.

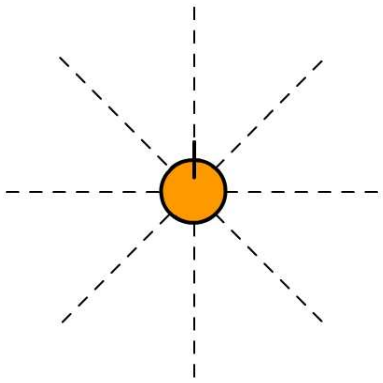
V ďalšom kroku sme zväčšili fitness protihráčov v pomere ku fitness spoluhráčov. Tým sme zabezpečili, že tí spoluhráči, pri ktorých bolo viac protihráčov, neboli vybraní pri prihrávaní.

Tiež sme odhalili chybu v implementácii v rozdelení lúčov v priestore okolo hráča. Túto chybu sme opravili.

5.1.2 Prihrávky brankára

Brankárovi tímu Gang of Six sa pomerne často stavalo, že zvolil prihrávku súperovmu hráčovi. Čiastočne sa tento problém podarilo vyriešiť zmenou vyrátavania užitočnosti jednotlivých alternatív prihrávok. Táto zmena je opísaná v predchádzajúcej kapitole.

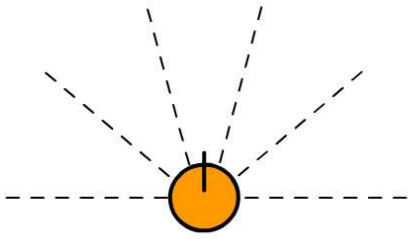
Tím Gang of Six používal jeden algoritmus prihrávania pre každého hráča tímu. Teda aj brankár aj hráči v poli používali rovnaký algoritmus vyrátavania najlepšej prihrávky. Hráč pri rozhodovaní sa, komu prihrať, si rozdelí priestor na ihrisku do niekoľkých lúčov. Všetky lúče spolu musia tvoriť samozrejme celý priestor okolo hráča. Situácia, keď si hráč rozdelí priestor okolo seba do lúčov je znázornená na Obr. 4.



Obr. 4 Lúče hráča pri prihrávaní

Hráč na obrázku si rozdelil hraciu plochu okolo seba na 8 úsekov. Následne podľa algoritmu opísaného v predchádzajúcej kapitole si hráč vyberie priestor kam kopne loptu. Rovnaký algoritmus používa aj brankár. Tento prístup však nie je vhodný, lebo brankár nemôže prihrávať smerom dozadu. Zbytočne si tak rozdelil priestor na úseky aj za sebou, keďže tento priestor vôbec neprichádza do úvahy.

Túto časť sme vylepšili takým spôsobom, že brankár použije rovnaké množstvo lúčov ako ostatní hráči, ale len na rozdelenie priestoru pred sebou. Priestor brankára sa teda rozdelí na menšie úseky a brankár sa tak môže lepšie rozhodnúť kam prihrá. Na Obr. 5 je zobrazená rovnaká situácia ako na predchádzajúcom obrázku, ale teraz je zobrazeným hráčom brankár mužstva.



Obr. 5 Lúče brankára pri prihrávaní

Ako je možne vidieť, brankár už zbytočne nevyhodnocuje aj priestor za sebou. Naopak lúče, ktoré takto ušetril používa na detailnejšie rozdelenie priestoru pred sebou. Výhody tohto prístupu sú v tom, že brankár môže teraz kvalitnejšie vyhodnocovať situáciu pred sebou. Prihrávky brankára sú po vykonaní tejto zmeny skutočne úspešnejšie.

5.1.3 Výsledky našej implementácie

Podarilo sa nám viac vyladiť rozohrávanie brankára. V prípade, že všetci spoluhráči sú krytí

protihráčmi, nahrával náš vylepšený brankár presnejšie a bezpečnejšie.

Aj napriek tomu sa však stávalo, že brankár niekedy nahral priamo protihráčovi, pri ktorom nebol v blízkosti žiaden náš hráč. Kvôli komplikovanosti výpočtu fitness, aký zvolil tím Gang Of Six, sa nám nepodarilo odchaliť príčinu týchto chýb. Gang Of Six totiž svoju prvotnú implementáciu refaktorizoval a prispôsobil na použitie v koordinačných grafoch. Zaviedol osekávanie fitness na špecifický interval <0, 12>. Je možné, že kvôli týmto zmenám sa nahrávanie zhoršilo resp. vniesli tým do kódu nejakú chybu.

5.2 Vylepšenie pohybu brankára

V tejto časti sme sa zamerali na implementáciu bránenia a výber brankára. Zdokonalenie týchto vlastností a upravenie už existujúcich možností a algoritmov, ktoré sú k dispozícií z prevzatého hráča tímu Gang of Six. Zdokonalili sme a upravili sme existujúcu analýzu navrhovaného riešenia po dôkladnej konzultácii s pedagogickým vedúcim, ktorý nám pomohol odhaliť v nami navrhovanom riešení v projektovej dokumentácii. Upravili sme existujúce riešenia a odladili sa drobné chyby v logike bránenia. Rozpracovali sme dva základné prístupy bránenia hráča v roli brankára tímu.

1) Prvý je založený v zmene konštanty v implementovanom riešení
2) Úprava výbehu brankára a jeho vracanie sa späť do bránkoviska

Aby sme vedeli správne odhadnúť chyby už implementovaného riešenia rozhodovania sa brankára, tak sme podrobne rozanalyzovali logiku existujúceho rozhodovania sa brankára a krok po kroku sme uvažovali, či zrealizované riešenie je korektné, resp. konfrontovali sme ich riešenie s našimi znalosťami o bránení. Na základe získaných informácii sme schopní uskutočniť i väčšie zmeny v celkovej logike brankára a nie len parciálnej časti.

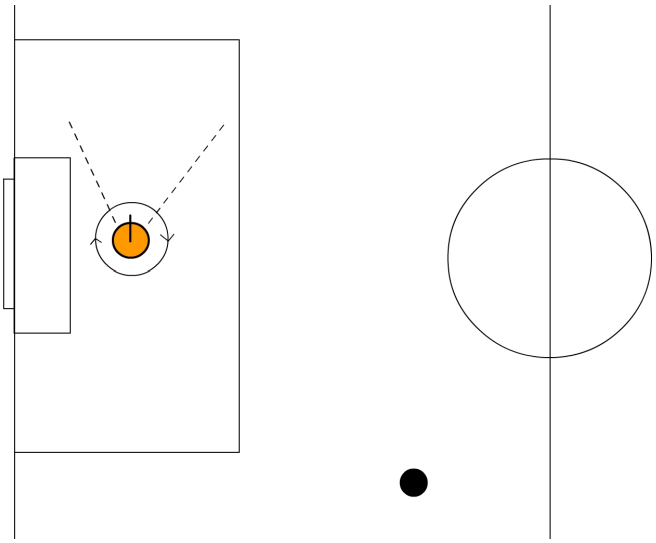
V nasledujúcich riadkoch stručne vovedieme čitateľa do problematiky priblížením funkčnosti a logickej následnosti krokov existujúceho algoritmu, aby ľahšie mohol pochopiť zmeny, ktoré sme implementovali v predkladanom riešení.

5.2.1 Zmena konštanty

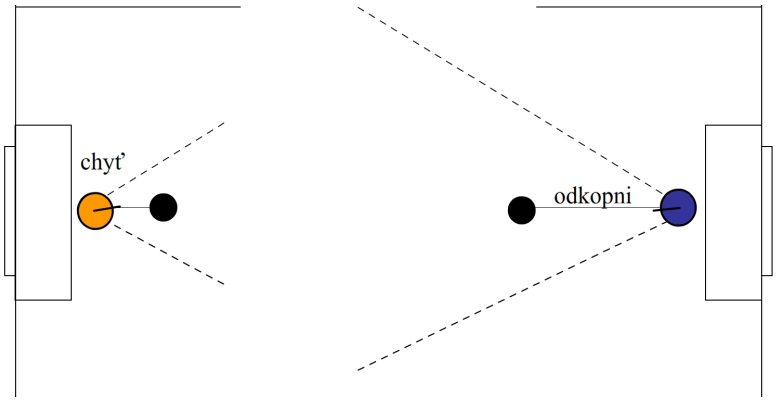
Po dôkladnej a opätovnej analýze implementovaného kódu logiky bránenia, sme sa rozhodli, že v prvotnej fáze zasahovania do kódu brankára zmeníme len určité logické nedostatky, ktoré by mohli viesť k lepšiemu bráneniu a navrhneme ďalší postup práce v nasledujúcom semestri.

Logika brankára je zrealizovaná v triede `PlayerTeams.cpp` v metóde `deMeer5_goalie`, ktorá je v každom cykle volaná a na základe získaných informácii z okolitého sveta a informácii, ktoré brankár má možnosť získať sám, sa rozhoduje o svojich krokoch. V prvotnej fáze rozhodovania brankára o svojom nasledujúcom konaní, sa zisťuje, či hra je v stave pred prvotným výkopom lopty. Ak áno overí sa, či je mužstvo, resp. brankár na svojej strategickej pozícií. Na základe toho sa buď brankár nastaví na strategickú pozíciu alebo prípadne sa len natočí smerovanie jeho tela na bod rozohratia. V ďalšom kroku sa kontroluje vlastnosť viditeľnosti lopty. To znamená, či brankár vidí loptu. Ak nie, tak sa pokúša ju nájsť (viď Obr. 6), inač pokračuje overovaním ďalších možných krokov, ktoré by mohol brankár vykonať. Nasleduje overenie aktuálneho stavu sveta z cieľom dosiahnuť aspoň jeden z nasledujúcich stavov `WM->getPlayMode() == PM_PLAY_ON, WM-`

`>isFreeKickThem()` alebo `WM->isCornerKickThem()`. Po splnení aspoň jednej zo spomínaných podmienok nasleduje rad ďalších kontrol možnosti operácií, ktoré by mohol brankár vykonať. V prvom rade sa overí, či nie je možnosť loptu chytiť funkciou `WM->isBallCatchable()`. Ak áno, pošle sa príkaz na vykonanie operácie chytenia. V prípade nemožnosti uskutočnenia spomínanej operácie sa overuje nasledujúca možnosť odkopu lopty (viď Obr. 7). V prípade možnosti nasleduje odkop lopty preč od bránky. Nespôsobilosť vykonania tejto operácie spôsobená aktuálnym stavom sveta má za následok kontrolu a overovanie ďalších možnosti. V predposlednej podmienke sa overí, či sa lopta nachádza v šestnástke, a zároveň či prvý pri lopte môže byť brankár. Ak podmienka je splnená, tak brankár zaujme pozíciu bránenia a otočí krk smerom k lopte. Ak by vzdialenosť medzi aktuálnou pozíciou lopty a brankárom bola menšia ako určená konštanta a brankár by bol najrýchlejším hráčom, tak brankár vybehne za loptou. Ak nie je splnená žiadna z vyššie uvedených podmienok, tak sa vytvorí priamka medzi stredom bránky a loptou. Brankár prostredníctvom tejto priamky natáča celé svoje telo smerom k lopte, a tak obraňuje.



Obr. 6 Hľadanie lopty brankárom

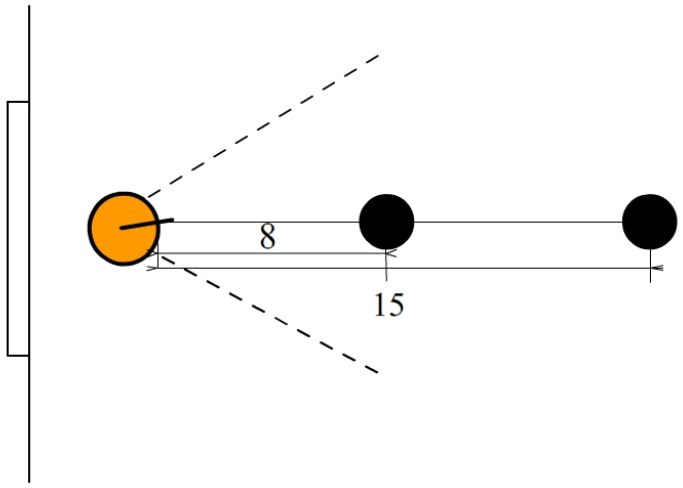


Obr. 7 Abstraktná ukážka vzdialenosti pre chytenie a odkop lopty

Následne sme analyzovali poslednú podmienku, ktorá sa nachádza v logickej následnosti krokov bránenia brankára. Zamerali sme sa na testovanie preddefinovanej konštanty a rozpracovali sme možnosť jej prípadnej zmeny. Navrhovaná výmena aktuálnej hodnoty za novú je podložená a overená množinou testov, ktoré sa vykonali po zmene konštanty. Samotné testy pozostávali zo simulovania možných stavov, ktoré môžu nastať počas zápasu. Na základe získaných výsledkov sme sa rozhodli pristúpiť k výmene a rozšíriť ju na konštantu s hodnotou 15 z pôvodných 8 viď (Obr. 8). Táto zmena rozšíri množinu výbehov brankára o ďalšie možnosti a stavy, avšak zachová sa podmienka bezpečnostného výbehu za loptou. To znamená, že aj keď brankár môže vybehnúť za vzdialenejšou loptou, vždy je to bezpečná vzdialenosť, v ktorej je isté, že sa k nej brankár skôr dostane ako súperov hráč. Predpokladáme, že sa nám podarilo rozšíriť množinu úspešných zásahov brankára. A tým eliminovať nežiadúce situácie, ktoré mohli nastať statickým vyčkávaním v bránke na loptu.

Časť kódu s implementovanou zmenou popisovanej konštanty:

```
if(((WM->getAgentGlobalPosition()  -  WM->getGlobalPosition(  OBJECT_BALL
)).getMagnitude() < 15 )){……
……
}
```

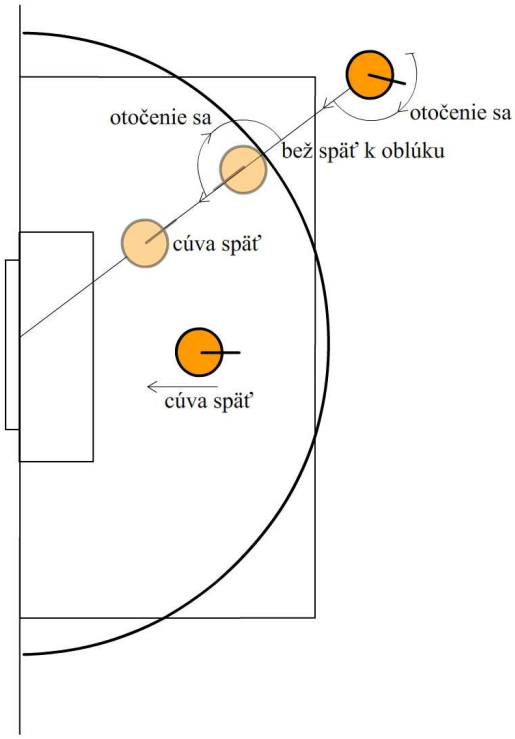


Obr. 8 Zmena výbehu brankára

5.2.2 Úprava návratu brankára do brániacej pozície

V časti hrubého návrhu sme podrobne opísali nové metódy návratu brankára na jeho aktuálnu pozíciu. Opísali sme možne stavy, ktoré môžu nastať a najvhodnejšie riešenia danej situácie. Následne sme sa venovali podrobnej analýze kódu a konfrontácii názorov s pedagogickým vyučujúcim, ktorý nás upozornil na situácie a stavy, ktoré náš prvotný návrh nebral v úvahu. Medzi hlavné nedostatky patrilo to, že je rozdiel byť v šestnástke v pravom hornom rohu ako priamo na vrchu pred bránkou. Tým sme chceli povedať, že je tam rôzna vzdialenosť návratu na domovskú pozíciu. Ďalším nedostatkom bola možnosť otočenia sa pred šestnástkou a dobehnutie do nej, kedy sa nebralo v úvahu, či pred brankárom sa nenachádza protihráč. Tým by mohla vzniknúť situácia, že sa brankár otočí protihráčovi chrbtom. Po dôkladnom premyslení a spracovaní nových poznatkov sme dospeli k záveru prepracovania prvotného

možnosti riešenia daného problému. Avšak samotná myšlienka rozdelenia spôsobu návratu brankára na dve spôsoby na základe vzdialenosti od bránky bola zachovaná. V prvotnej fáze sa uvažovalo o tom, či sa brankár nachádza v šestnástke alebo už je mimo nej. Podľa toho sa vyberal spôsob návratu. V aktuálnom vylepšení sa kontroluje vzdialenosť od bránky, strategickej pozície v tvare kruhu (viď Obr. 9). To znamená, že sa kontroluje, či brankár je od danej pozície vzdialený istý počet metrov. Odstúpilo sa od kontroly, či sa nachádza v šestnástke. A zároveň sa vložila podmienka, či sa v jeho blízkosti nenachádza protihráč. Ak sú splnené obe požiadavky, vtedy sa vykoná druhý spôsob návratu do bránky, pozostávajúci z otočenia a dobehnutia do opisovaného kruhu a spätné otočenie a následné cúvanie. Ak je porušená aspoň jedná z nasledujúcich požiadaviek, brankár sa vracia na svoju pozíciu prvotne opísanou metódou v hrubom návrhu.



Obr. 9 Úprava výbehu brankára

Z dôvodov rozsiahlosti implementácie tejto úpravy vlastnosti brankára nebolo možne z časového hľadiska doimplementovať a hlavne otestovať celú funkčnosť v rámci prototypu. Takže celková implementácia bude súčasťou finálneho hráča.

6 Spustenie servera a build hráčov

6.1 Server a monitor

Pre potreby simulácie a testovania je potrebné mať nainštalované nástroje, ktoré umožnia spustiť prostredie, kde sa odohráva zápas, (soccer server) a monitor, ktorý vizualizuje zápas. Produkt sme testovali s nástrojmi:

- Robocup Soccer Server (verzia 9.4.5) – server.
- Soccermonitor (verzia 1.4) – monitor na vizualizáciu zápasu.

Tieto nástroje budú v dostupne na stiahnutie na stránke tímu. Pre ich spustenie je potrebné:

- Siahnuť skomprimovaný archív so súbormi.
- Rozbaliť archív.
- Pre spustenie servera je potrebné spustiť súbor *rcssserver.exe* nachádzajúci sa v priečinku servera.
- Pre spustenie monitora je potrebné spustiť súbor *Soccermonitor_localhost.exe* v adresári monitora.

6.2 Hráč

Pre kompiláciu zdrojových súborov hráča je potrebná sada nástrojov MinGW

(http://www.mingw.org), ktorá obsahuje prekladač gcc, program make a potrebné knižnice. Po kompilácii vzniknú súbory *player.exe* a *coach.exe*. Spustením týchto súborov spustíme hráča a kouča. Tieto súbory pri spúšťaní vyžadujú mať nastavené parametre. Hráč parametre: číslo_hráča, názov_tímu, umiestnenie_servera; kouč má parametre: názov_tímu, umiestnenie_servera.

6.3 Simulácia zápasu

Pre uľahčenie simulácie zápasu je v adresári build vytvorený skript *start-all.bat*. Opis činnosti tohto skriptu je v kapitole Spúšťacie skripty.

V adresári build sa taktiež nachádzajú skripty:

- start-coach.bat – pre spustenie kouča s tímom Jahodoví princovia.
- start-team.bat – spustenie tímu Jahodoví princovia.
- start-team2.bat – spustenie tímu, proti ktorému sa hra zápas.

Pre spustenie simulácie zápasu je potrebná mať spustený server, monitor a hráčov z obidvoch tímov. Monitor poskytuje grafické používateľské rozhranie, kde je potrebné kliknúť na tlačidlo výkop (ikonka s futbalovou loptou). Potom sa začne hrať zápas. Po polčase je opätovne potrebné kliknúť na tlačidlo výkop.

Simuláciu najjednoduchšie vypneme stlačením kombinácie kláves Crtl+C v okne soccer servera.

7 Spúšťacie skripty

Pre pohodlné testovanie a opakované spúšťane simulácií sme si vytvorili skripty, ktoré nám umožnili zautomatizovať viaceré činnosti. Po viacerých implementáciách sme dospeli ku skriptu, ktorý vykonáva:

- Zmaže staré logovacie súbory.
- Spustí server.
- Spustí SoccerMonitor.
- Spustí momentálne vyvíjaného hráča.
- Spustí kouča momentálne vyvíjanému hráčovi.
- Spustí 2. tím, proti ktorému sa hra zápas.

Momentálne sa ako 2. tým, proti ktorému sa hrá zápas, používa hráč, s vlastnosťami, ktoré získal od tímu Gang of Six. Týmto skriptom po vytvorení buildu súčasného hráča spustíme zápas proti inému tímu. Tento skript veľmi uľahčil testovanie a pomocou neho zaznamenávame výraznú časovú úsporu, ktorú by si inak vyžadovalo ručné spustenie všetkých prvkov v systéme s ich parametrami. V súčasnosti stačí spustiť skript (je to súbor typu *.bat), počkať chvíľu, kým sa spustia všetky prvky systému, a potom iba v monitore kliknúť na ikonku *Výkop*.

8 Záver

Prototyp prináša so sebou prvotné pokusy o implementovanie nosných časti z hrubého návrhu riešenia. Jednotlivé časti implementácie boli počas vývoja testované na množine rôznych testov pokrývajúcich oblasť riešenej problematiky.

Analýza a následná zmena konštanty výbehu brankára umožnila rozšíriť množinu úspešných zásahov a zároveň si zachovať nezmenený stav úspešnosti bránenia. Zrýchlila sa efektivita a rýchlosť hry brankára. Nie je už takou statickou časťou hry, ako bol pred samotnou zmenou. Pripravili sa podklady pre následnú implementáciu zdokonalenie vo vracaní sa na aktuálnu pozíciu.

Podarilo sa osvojiť si vlastnosti a možnosti, ktoré poskytuje loganalyzer, ktorý nám uľahčil prácu pri analyzovaní viacerých sporných situácií.

Zdokonalili sme rozohrávanie brankára. Zdokonalenie sa nachádza a je súčasťou prototypu. Aj táto časť bola dôkladne otestovaná.

V samotnom závere sa podarilo implementovať kauča a jeho vlastnosti, ktoré sa budú ďalej využívať v prospech hráča. Je možne, že sa bude kauč a jeho vlastnosti v nasledujúcom semestri rozširovať.

Príloha A - Zoznam použitých obrázkov

Príloha B - Zoznam použitých tabuliek