Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný odbor: Softvérové inžinierstvo

Robocup – tretí rozmer
Tímový projekt

Tím č. 1 - Hazard Tím:
Bc. Vladimír Krivuš, Bc. Peter Mešťaník, Ing. Andrej Neczli,
Bc. Ján Suchal, Bc. Peter Vojtek, Bc. Peter Židek

Vedúci projektu: Ing. Marián Lekavý školský rok: 2005/2006

Obsah

Úvod

Tento dokument bol vytvorený v rámci predmetu Tímový projekt a venuje sa virtuálnej simulácii 3D robotického futbalu pod názvom RoboCup 3D.

Dokument na skladá z viacerých častí.

Časť analýza sa venuje analýze platformy pre simuláciu, tzv. 3D servera. Ďalej sa v analýze nachádza zhodnotenie výsledkov práce tímov z predošlých rokov, pričom pozornosť sa venuje domácim (fakultným) aj zahraničným tímom, 2D aj 3D simulácii.

V špecifikácii požiadaviek určujeme naše požiadavky a ich rozsah. Tiež sú tu vymedzené oblasti, ktorým sa budeme venovať.

Hrubý návrh popisuje základy architektúry hráča. Keďže nenadväzujeme na výsledky predošlých tímov, venujeme sa aj návrhu samotného prostredia, v ktorom sa bude hráč vyvíjať.

Dokumenty v prílohe predstavujú najmä údaje a dokumenty súvisiace s riadením, koordináciou a komunikáciu v našom tíme.

1.1 Simulačná liga RoboCup

Projekt RoboCup zastrešuje na celosvetovej úrovni organizácia RoboCup Federation, ktorá sa venuje podpore a propagácii robotiky a umelej inteligencie. Nosnou myšlienkou je pomocou umelých prostriedkov vytvoriť hráča, ktorý by sa svojimi schopnosťami čo najviac blížil ľudským hráčom.

RoboCup Federation každoročne organizuje medzinárodný turnaj, ktorý má dve hlavné časti, robotickú a simulačnú.

Simulačná liga RoboCup sa orientuje na softvérové agenty, hrajúce futbal podľa upravených pravidiel v 2D a 3D priestore. Táto časť súťaže sa nevenuje hardvérovej stránke problému, pozornosť je sústredená len na rozvíjanie inteligencie agentov. Vzhľadom na zložitosť systému a snahu priblížiť sa k schopnostiam reálnych hráčov sa využívajú najmä poznatky z umelej inteligencie.

2D simulácia je už dostatočne rozvinutá a jej pravidlá sú stanovené tak, aby čo najvernejšie zodpovedali reálnemu futbalu. 3D simulácia je ešte len na svojom začiatku a preto zatiaľ nie je natoľko prepracovaná.

NA FIIT (FEI) STU sa projekt RoboCup rozvíja od roku 1999 v rámci viacerých predmetov. Na fakulte sa pravidelne koná turnaj vo futbale, ktorý slúži ako porovnanie výsledkov jednotlivých tímov. Všetky tieto snahy sa zatiaľ upriamovali len na 2D simuláciu, náš projekt – 3D simulácia – na fakulte nemá predchodcu.

1.2 Zadanie projektu – Robocup – tretí rozmer

Téme RoboCup, presnejšie lige simulovaného robotického futbalu sa naši študenti venujú už šesť 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 dosť popísané aj na stránke STU turnaj v simulovanom robotickom futbale.

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

Hlavným cieľom projektu bude vytvoriť hráča pre 3D simuláciu, ktorý umožní ďalším tímom pokračovať v tejto práci. Keďže pôjde o vytvorenie novej platformy pre ďalší vývoj, základnými požiadavkami sú prehľadnosť a ďalšia rozšírovateľnosť, a to na úrovni návrhu aj implementácie. Dôraz pri vytváraní hráča by mal byť kladený na dobre prepracované a odladené nižšie schopnosti hráča, ktoré umožnia hráčovi spracovávať vnemy z prostredia a efektívne konať v prostredí (pohybovať sa, pracovať s loptou). Hoci pôjde o vytvorenie nového hráča, bude možné pri návrhu a implementácii čerpať z veľkého množstva prístupov existujúcich v 2D simulácii. Zimný semester je vyhradený na oboznámenie sa s celým simulačným prostredím, a takisto s existujúcimi hráčmi, ďalej návrhu a prototypovej realizácii hráča. Súčasťou bude aj vytvorenie plánu implementácie a overovania prístupu v nasledovnom semestri. V letnom semestri nás čaká podrobné vypracovanie návrhu a jeho implementovanie a overovanie. Nemenej podstatnou časťou projektu bude vytvorenie dokumentácie, ktorá poskytne tímom v ďalších rokoch odrazový mostík pri použití vytvoreného hráča.

2 Analýza

Analýza sa skladá z analýzy 3D servera na jednej strane a analýzy existujúcich tímov a ich hráčov na strane druhej.

2.1 Analýza 3D servera

RCSSServer 3D je fyzikálne simulačné prostredie. Jeho prvotnou úlohou je poskytnúť „univerzálne“ simulačné prostredie pre rôzne druhy simulácií. Aby mohol spĺňať túto funkciu, hlavné časti simulátoru sú koncipované ako plugin moduly. Takéto usporiadanie umožňuje jednoduchú obmenu jednotlivých modulov, pričom pri použití skriptovacieho jazyka Ruby je možné jednotlivé moduly meniť bez nutnosti rekompilácie modulov.

2.1.1 Implementácia RCSSServer 3D

V nasledujúcej stati popíšeme najdôležitejšie knižnice vytvárajúce simulátor RCSSServer 3D.

Knižnica zeitgeist

Knižnica zeitgeist bola vytvorená za účelom poskytnutia mechanizmu na prácu s objektami. V ponímaní knižnice zeitgeist trieda poskytuje rozhranie na vytváranie svojich inštancií. Okrem tohto mechanizmu je v knižnici zeitgeist implementovaný aj mechanizmus vytvárania hierarchie objektov. Táto hierarchia je reprezentovaná virtuálnym súborovým systémom, kde jednotlivé „adresáre“ a „súbory“ sú inštanciami tried. Všetky objekty v hierarchii sú identifikované jedinečným menom, „cestou“ v súborovom systéme. Okrem týchto mechanizmov poskytuje knižnica ďalšie dôležité služby prístupné vo forme serverov v hierarchii objektov a to LogServer, FileServer, ScriptServer.

Knižnica oxygen

Knižnica oxygen predstavuje momentálny simulačný engine. Tento engine je implementovaný s využitím knižnice zeitgeist. Medzi poskytované služby engine-u patrí PhysicsServer, AgentAspect, ControlAspect a SceneServer. PhysicsServer je zodpovednú za fyzikálnu stránku simulovaného prostredia. AgentAspect definuje schopnosti agenta, zatiaľ čo ControlAspect určuje podmienky simulácie. SceneServer je zodpovedný za simuláciu virtuálneho sveta.

Knižnica kerosin

Knižnica kerosin poskytuje vizuálny aspekt simulácie jej súčasťou sú ImageServer, FontServer, OpenGLServer, TextureServer, MaterialServer. Okrem týchto používaných služieb poskytuje knižnica kerosin aj služby pre interaktívne simulácie - InputServer, SoundServer.

Ako sme už v úvode spomenuli, server je modulárny systém. V ďalšom texte sa preto zameriame na krátky opis jednotlivých modulov, ktoré umožňujú vytvoriť trojrozmernú fyzikálnu simuláciu.

Perceptor

Perceptor je objekt, ktorý umožní agentom vnímať vnemy zo simulovaného prostredia. Všetky novoimplementované perceptory musia byť odvodené od triedy `Kerosin::Perceptor`. Novovytvorený perceptor musí mať implementovanú jedinú funkciu a to `bool Percept(TDictionary & dictionary)`. Trieda TDictionary je hash tabuľka, kde primárnym kľúčom do tabuľky je reťazec. Z dôvodu tejto flexibility musí mať každý perceptor definované výstupy vo forme kľúč – typ výstupnej hodnoty, aby sa predišlo chybám.

Effector

Effector je objekt, ktorý umožňuje agentovi ovplyvňovať svoje okolie v simulovanom prostredí. Novovytvorený effector musí mať implementovanú jedinú funkciu a to `bool Perform(boost::shared_ptr<BaseNode> &base, float deltaTime)`, kde prvý parameter predstavuje objekt, ktorý bude ovplyvnený pôsobením effectora a druhý parameter predstavuje dĺžku simulovaného času, ktorý „spotrebuje“ effector počas svojej aktivity.

ControlAspect

ControlAspect je objekt zodpovedný za dodržiavanie pravidiel simulácie, jeho ďalšou úlohou je určovanie, ktorý agent má nárok na ktorý perceptor, alebo effector.

AgentAspect

AgentAspect je objekt vytvárajúci agenta, pomocou perceptorov získava informácie o okolitom simulovanom prostredí, pomocou effectorov toto prostredie ovplyvňuje. Okrem interakcie s okolím je agent schopný „myslieť“.

2.1.2 Simulácia futbalu s použitím RCCSServer 3D

V tejto časti sa zameriame na krátky opis simulácie futbalu s využitím platformy RCSSServer 3D.

Parametre objektov simulácie

Ihrisko je rovina s rozmermi dodržiavajúcimi FIFA štandardy, dĺžka 100 – 110 m, šírka 64 – 75 m. Dĺžka brány je určená na 7,32 m, výška brány je 2,44 m. Momentálne je vo futbalovej simulácii obmedzená výška brány na 0,5 m. Lopta je guľa s polomerom 0,222 m a hmotnosťou 0,41 – 0,45 kg.

Agenti sú v prostredí simulácie reprezentovaný ako gule s priemerom 0,44 m a hmotnosťou 75 kg. Všetci agenti v simulácii majú rovnaké fyzikálne vlastnosti.

Interakcia s prostredím

V tejto časti sa zameriame na popis prostriedkov pomocou, ktorých dokáže agent získavať informácie o simulovanom prostredí a na popis prostriedkov pomocou, ktorých dokáže toto prostredie ovplyvňovať.

Energia agenta

Každý agent je vybavený rovnakou samodobíjacou batériou, rôzne úkony túto batériu rôzne vyčerpávajú. V prípade, že je agent bez energie nemôže žiadnym spôsobom ovplyvňovať okolie a musí počkať kým sa mu energia dobije.

Pohyb po ihrisku

Každý agent je vybavený pohonnou jednotkou, ktorá mu umožňuje pohybovať sa všetkými smermi. Táto pohonná jednotka funguje iba v prípade, že sa agent dotýka zeme. V prípade vypnutia jednotky agent okamžite nezastaví ale pokračuje určitú dobu v pohybe predchádzajúcim smerom, jednotku je v takomto prípade možné použiť na brzdenie. Použitie tejto pohonnej jednotky je prístupné agentom pomocou driveeffector-u. Jediným vstupným parametrom tohto effector-u je trojrozmerný vektor s maximálnou dĺžkou 100 jednotiek, kladný smer x-ovej osi smeruje vždy smerom ku protivníkov a kladný smer z-ovej osi smeruje vždy smerom nahor. Agent dáva príkaz na použitie pohonnej jednotky pomocou reťazca, napr.:

```
(drive 20.0 50.0 0.0).
```

Na pohon pohonnej jednotky využíva agent energetické zásoby z batérie.

Práca s loptou

Existujú dva spôsoby ovplyvňovania pohybu lopty hráčom. Prvým je beh s loptou, ktorý je realizovaný použitím pohonnej jednotky agenta na postrkovaním lopty požadovaným smerom pomocou driveeffector-u. Druhým spôsobom je kopnutie do lopty, tento spôsob je realizovaný pomocou kickeffector-u. Vstupnými parametrami kickeffector-u sú vertikálny uhol v rozmedzí od 0 – 50° a určením sily v rozmedzí 0 – 100 % dostupnej sily na základe stavu batérie. Volanie kickeffector-u je realizované pomocou reťazca v tvare napr.: `(kick 20.0 80.0)`

Vytvorenie agenta

Pri prvom pripojení na simulátor nemá žiadny agent v simulácii vytvorenú svoju reprezentáciu. Preto je potrebné hneď na úvod zavolať createeffector. Pretože v súčasnosti ešte nie sú implementované všetky požadované vlastnosti simulácie, volanie createeffector-u je obmedzené iba na reťazec v tvare `(create)`. Po vytvorení agenta pomocou createeffector-u je nutné tohto agenta priradiť k jednému z tímov a takisto je nutné priradiť mu číslo dresu, ktoré sa spolu s menom tímu stane jeho identifikátorom. Tieto priradenia sú realizované pomocou initeffector-u reťazcom v tvare: `(init (unum 7) (teamname HazardTeam))`.

Vnímanie sveta

Agent je schopný vnímať entity simulovaného sveta pomocou visionperceptor-u. Rozhľad agenta je definovaný na 360°, čiže vidí všetkými smermi. Výstup visionperceptor-u sú informácie o objektoch prostredia. Objektmi prostredia sú napríklad ostatný agenti, lopta, alebo niektoré značky na ihrisku, ako napríklad stred ihriska, rohová zástavka a podobne. O každom z týchto objektov poskytne visionperceptor tri informácie - vzdialenosť, horizontálny a vertikálny uhol k tomuto objektu. Všetky tieto hodnoty sú počítané relatívne k polohe a smeru agenta, ktorý volal visionperceptor. Výstup visionperceptoru bude tvoriť napríklad takýto reťazec:

```
(Vision (Flag (id 1_l) (pol 54.3137 -148.083 -0.152227)) (Flag (id 2_l) (pol
59.4273 141.046 -0.131907)) (Flag (id 1_r) (pol 61.9718 -27.4136 -0.123048))
(Flag (id 2_r) (pol 66.4986 34.3644 -0.108964)) (Goal (id 1_l) (pol 46.1688
179.18 -0.193898)) (Goal (id 2_l) (pol 46.8624 170.182 -0.189786)) (Goal (id
1_r) (pol 54.9749 0.874504 -0.149385)) (Goal (id 2_r) (pol 55.5585 8.45381 -
.146933)) (Ball (pol 6.2928 45.0858 -0.94987)) (Player (team robolog) (id 1)
(pol 7.33643 37.5782 5.86774)))
```

Zisťovanie stavu hry

Na zistenie stavu hry bol implementovaný samostatný perceptor s názvom gamestateperceptor. Jeho úlohou je podávať informácie o stave hry. Prvé informácie, ktoré tento perceptor poskytne, sú informácie o rozmeroch ihriska.

Stav agenta

Na zistenie stavu agenta bol implementovaný agentstateperceptor, v súčasnej dobe sa však tento perceptor obmedzuje na poskytovanie informácií o stave batérie agenta.

2.1.3 Vzorová implementácia hráča – agenttest

V tejto stati bude popísaná architektúra jednoduchého hráča dodávaného s RCSS Serverom.

Popis architektúry

Základ architektúry tohto jednoduchého hráča tvoria moduly:

- modul modelu sveta – worldmodel
- modul komunikácie so serverom – commserver
- modul hráča – kicknrun

Komunikácia so serverom

Komunikácia so serverom je zapuzdrená v triede, ktorá zodpovedá za prijímanie a posielanie správ serveru. Hráč komunikuje so serverom pomocou dvoch deskriptorov. Jeden z deskriptorov je určený na posielanie správ serveru, druhý slúži na prijímanie správ zo serveru. Každá zo správ, prichádzajúca aj odchádzajúca, je prefixovaná so svojou celkovou dĺžkou, maximálna dĺžka správy je 8192 znakov.

Model sveta

Slúži na sprostredkovanie informácii zo servera hráčovi. Model sveta je zapuzdrený v triede, ktorej hlavnou úlohou je prevod informácií poskytnutých perceptormi servera (Visionperceptor a Gamestateperceptor) do štruktúr a objektov zrozumiteľných pre agenta.

Modul hráča

Modul hráča je implementovaný vo viacerých úrovniach. Prvá úroveň obsahuje primitívnu funkcionalitu. V tejto úrovni agent získava funkcionalitu na spojenie s modulmi modelu sveta a komunikácie so serverom. V druhej úrovni sú implementované metódy na vyvolávanie effectorov. Každý effector poskytovaný pri simulácii futbalu je zapuzdrený vo vlastnej metóde. V tretej úrovni je implementované správanie sa agenta, ktoré je odvodzované od stavu hry.

2.2 Analýza riešení iných tímov

Súčasťou našej analýzy bola aj analýza tímov z minulých rokov.

Tieto tímy možno rozdeliť nasledovne:

- 2D tímy:
  - tímy FEI/FIIT STU BA:
    - Stjupit dox (2002)
    - Deravá kopačka (2002)
    - SKLO (2003)
    - FC Farmári (2004)
  - svetové tímy:
    - CMUnited-99 (1998)
    - UBU (1999)
    - UvA (1999)
- 3D tímy:
  - svetový tím:
    - Rolling Brains (2004)

Ako je z predchádzajúceho rozdelenia vidieť, na našej fakulte sa rieši RoboCup 3D po prvýkrát. Z toho vychádza aj zodpovednosť nášho tímu, a to vytvoriť návrh systému (ako aj systém samotný), ktorý by bol ďalej jednoducho rozširovateľný a čo možno najprehľadnejší. To tiež spôsobilo, že naše analýzy boli zamerané hlavne na návrhy hráčov jednotlivých tímov.

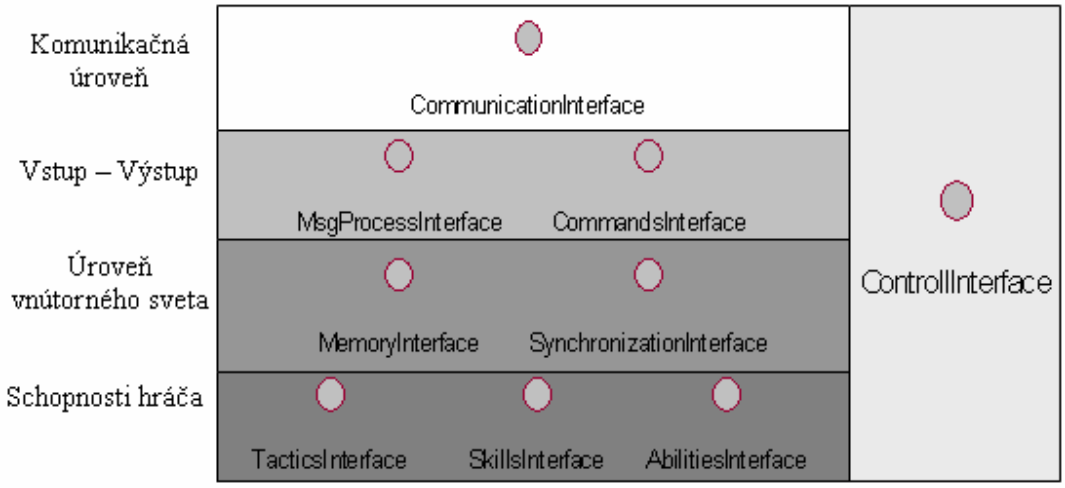
2.2.1 Stjupit Dox (2002)

Základné informácie

Tento tím bol vytvorený v roku 2002 a pochádza z návrhu tímu Roztoče z roku 2001.

Architektúra hráča vyzerá byť zvolená vhodne, pretože bez väčších zásahov ju využili autori tímu FC Farmári v roku 2004 a vyhrali lokálnu súťaž FIIT STU.

Návrh hráča



Obr. 1**.** Úrovne architektúry hráča tímu Stjupid Dogs.

Architektúra hráča sa delí na päť úrovní – komunikačnú, vstupno-výstupnú, úroveň vnútorného sveta, schopností a riadiaca úroveň.

Komunikačná úroveň zabezpečuje odosielanie, prijímanie a spracovávanie správ medzi hráčom a serverom. Vstupno-výstupná vrstva zabezpečuje abstrakciu správ pre vyššie vrstvy.

Úroveň vnútorného sveta zabezpečuje vytváranie, uchovávanie a zmenu vnútorného sveta spolu so spätnoväzobnými reakciami na výskyt určitých udalostí. Všetky informácie, ktoré hráč získa, sa uchovávajú v jeho vnútornom svete. Hráč čerpá informácie potrebné pri rozhodovaní práve z tohoto vnútorného sveta.

Úlohou úrovne schopností hráča je poskytnúť hráčovi základné a taktické schopnosti.

Tieto štyri úrovne navzájom prepája riadiaca úroveň, ktorá obsahuje hlavnú slučku programu hráča.

Za ďalšie zaujímavé vlastnosti z hľadiska architektúry hráča/tímu považujeme:

- možnosť vytvoriť kópiu aktuálneho stavu vnútorného sveta, ktorá sa využíva na trénovanie neurónovej siete ovládajúcej hráčov
- štatistické schopnosti kouča, ktorý podľa nich dokáže meniť globálnu stratégiu hry či samotného hráča

Výhody návrhu

- Vcelku stabilná a overená architektúra
- V lokálnej súťaži úspešný tím

Nevýhody návrhu

- Tím vytvorený pre 2D simuláciu

Zdroje

Horváth, Ivančo, Lacko, Pap, Trebatický, Kapustík: Simulácia robotického futbalu, tímový projekt. FIIT STU BA 2002.
http://www2.dcs.elf.stuba.sk/TeamProject/2002/team01/download/dokumentacia/FinalnaDokumentacia.zip

2.2.2 Deravá kopačka (2002)

Základné informácie

Tím deravá kopačka bol vytvorený v rámci predmetu Tímový projekt v akademickom roku 2002 / 2003 na Fakulte Elektrotechniky a Informatiky Slovenskej Technickej Univerzity. Vychádza z tímu z roku 2001 / 2002 "Roztoče". Jeho cieľom je vylepšiť inteligenciu prevzatého hráča. Výsledný tím sa zúčastnil školského turnaja, kde sa umiestnil na prvom mieste.

Návrh hráča

Architektúra hráča je ponechaná z pôvodného hráča. Jej štruktúra je uvedená na Obr. 2.1. Tieto triedy sú rozdelené do 3 vrstiev, z ktorých každá má iné schopnosti a vykonáva iné činnosti. Jednotlivé vrstvy majú za úlohu:

- **Dolná vrstva** Jej úlohou je zabezpečiť komunikáciu so serverom aj hráčmi, synchronizácie, spracovanie správ a tiež vytvára základný pohľad na svet.
- **Stredná vrstva** Má za úlohu vytvárať a vykonávať krátkodobé plány a zabezpečovať reaktívne správanie sa.
- **Horná vrstva** Je vrstva, ktorá zabezpečuje dlhodobé plánovanie a tieto plány posúva ako odporúčania strednej vrstve.

Zaujímavým riešením je použitie viacerých vlákien. Tieto boli pôvodne navrhnuté pre každú z horeuvedených vrstiev jedno, avšak neskoršou analýzou sa zistilo že postačujú dve vlákna. V jedno vlákne beží samostatne horná vrstva a v druhom bežia spoločne stredná a spodná vrstva. Tieto boli spojené do jedného vlákna, pretože boli na sebe priamo závislé a nebolo možné aby pracovali paralelne.

Výhody návrhu

Za hlavné výhody analyzovaného návrhu považujeme:
- Vhodné rozdelenie na 3 vrstvy
- Užitočné rozdelenie vrstiev do vlákien

Nevýhody návrhu

Za nedostatky analyzovaného návrhu považujeme:
- Časť inteligencie hráča by sa dal zabudovať priamo do štruktúry hráča, väčšie množstvo tried pre špecifickejšie situácie

Zdroje

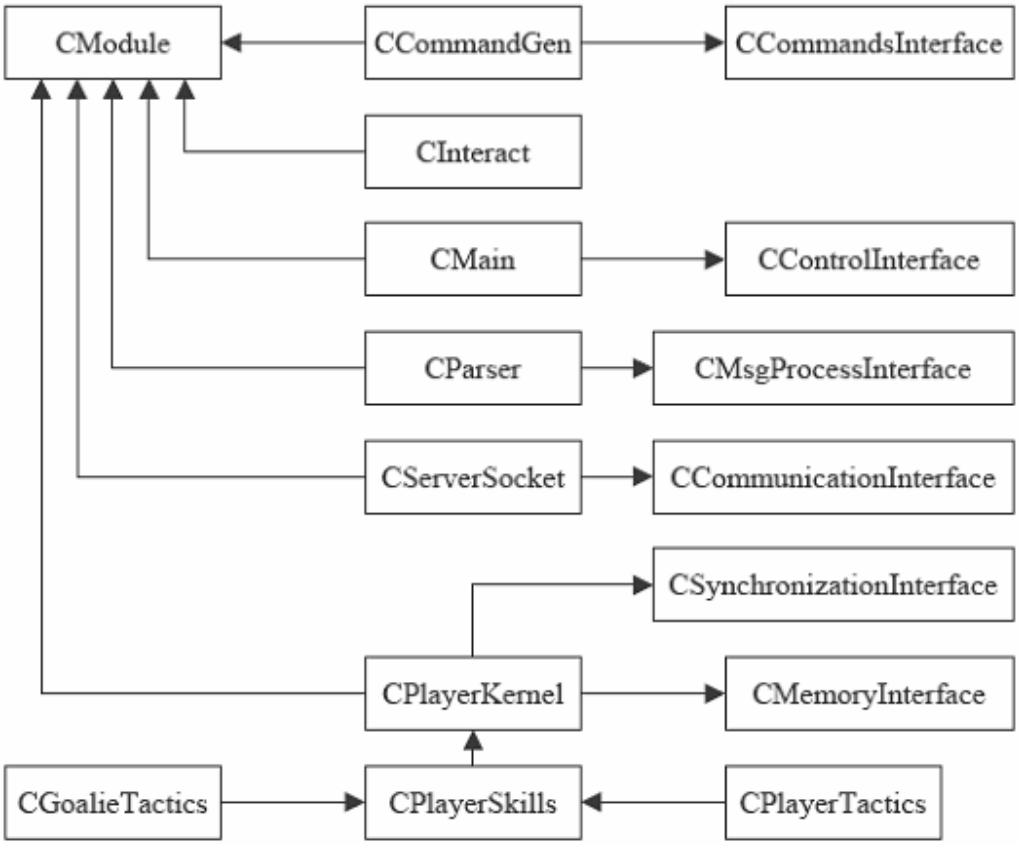
Kapusta R., Košík M., Košťál M., Lekavý M., Ulický M., Dirga P. : Deravá kopačka – softvér, 2003
http://www2.dcs.elf.stuba.sk/TeamProject/2002/team03/doc/softver.pdf

2.2.3 SKLO (2003)

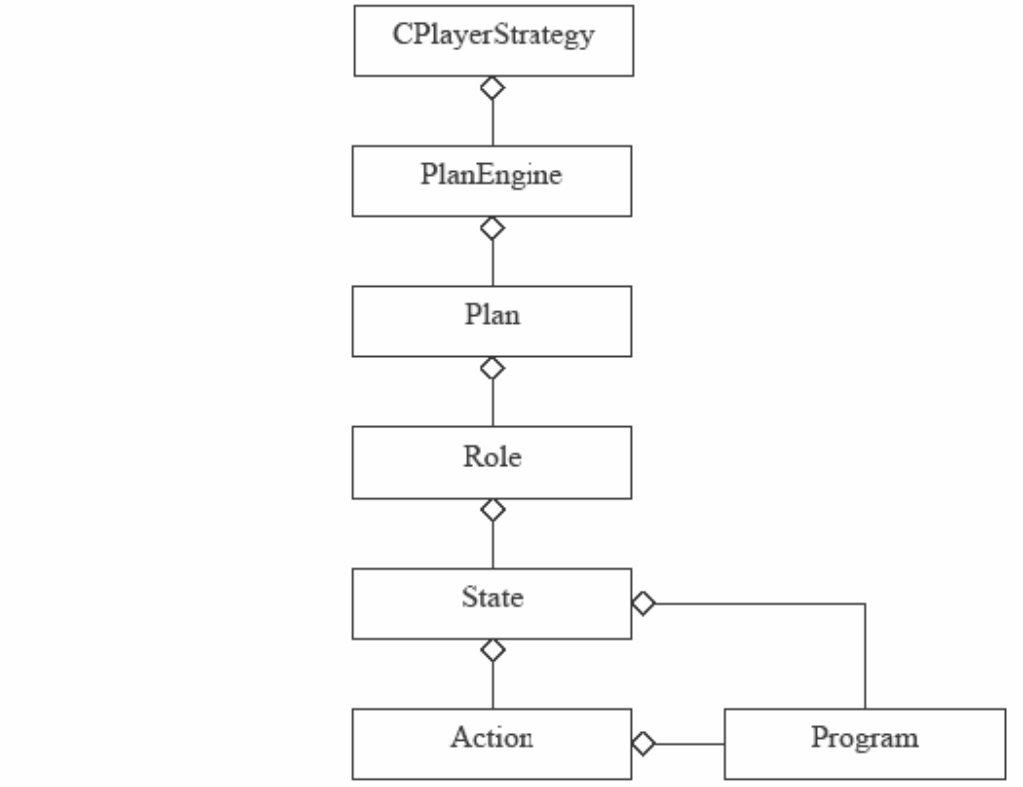
Základné informácie

Pôvod tímu: FIIT STU BA, Slovenská republika. Rok návrhu: 2003 Účastník: RoboCup FIIT 2004 – 2.miesto. Návrh vychádza z tímu Deravá kopačka.

Návrh hráča



Obr. 2. Architektúra hráča tímu SKLO.



Obr. 3**.** Strategická vrstva hráča tímu SKLO.

Výhody návrhu

- Zlepšenie spracovania vizuálnej informácie: prepínanie šírky pohľadu, otáčanie krkom, zlepšenie komunikácie medzi hráčmi formou nových signálov, ...

Nevýhody návrhu

Žiadne neboli identifikované.

Zdroje

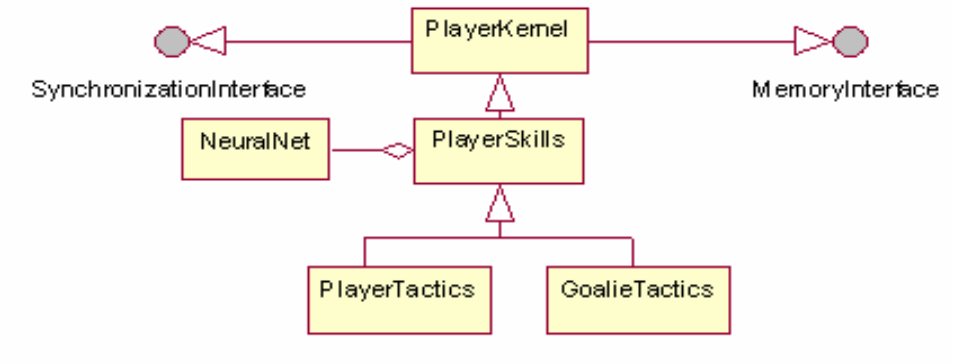
Gorbatchev, S., Križo, E., Mišík, P.: Tím SKLO, 2003.
http://www2.dcs.elf.stuba.sk/TeamProject/2003/team08/

2.2.4 FC Farmári (2004)

Základné informácie

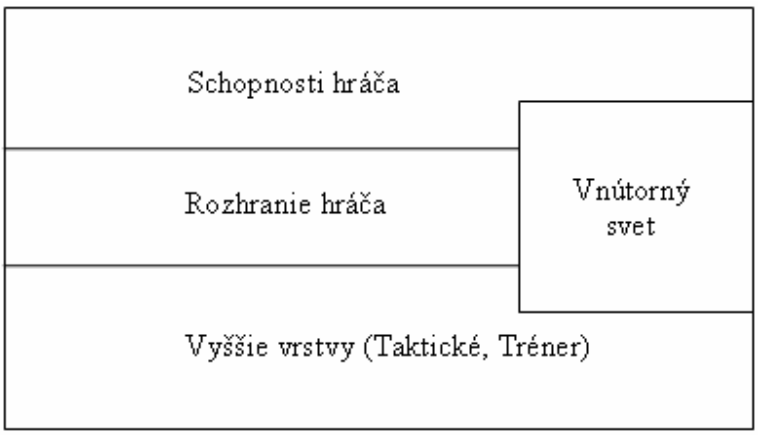
Tím FC Farmári vznikol v roku 2004 na FIIT STU. V roku 2005 sa zúčastnil fakultnej súťaže, ktorú vyhral. Návrh tímu vychádza z tímu z predošlého roku – Stjupid Dogs (návrh ktorého vychádza z tímu Roztoče). Práca tímu pozostávala okrem vylepšovania schopností hráča aj z implementácie kouča, ktorý vie analyzovať priebeh hry a dokáže hráčom dávať odporúčania. Tiež upravil a vylepšil systém formácií.

Návrh hráča



Obr. 4. Architektúra hráča tímu FC Farmári.

Návrh využíva neurónovú sieť na ovládanie nízkych schopností hráča. Presnejšie je použitá sieť typu *Feed Forward* sieť s metódou *Reiforcement Learning Temporal Difference Alfa* (RL-TD). Táto je využívaná na jednoduché akcie s loptou ako je driblovanie, prihrávanie, strela na bránu a pod.



Obr. 5. Oddelenie vrstiev hráča.

Vnútorný svet hráča sa tím FC Farmári rozhodol ponechať bez väčšej zmeny tak, ako ho prevzali od tímu Stjupid Dogs – presnejšie, tento pochádza ešte z návrhu tímu Roztoče.

Tím FC Farmári sa ďalej venoval rozvoju kouča, ktorý na základe pozorovania hry rozhoduje, akú formáciu hráčov na ihrisku použiť a tiež akou stratégiou hrať – sem spadajú aj špeciálne stratégie, ako extrémna obrana, extrémny útok.
Tím sa tiež venoval vytvoreniu nástroja pre zjednodušenie generovania kódu formácií hráčov.

Výhody návrhu

- použitie overeného prístupu
- schopnosti hráča preukázané víťazstvom na lokálnom turnaji

Nevýhody návrhu

- hráč je na príliš vysokej úrovni vzhľadom na naše ciele
- podobne, vyvíjané podporné nástroje sú zatiaľ pre nás zbytočné
- použitie neurónovej siete nebolo úspešné pri všetkých činnostiach, ktoré mala pokryť

Zdroje

Rusnák, M., Bernaťák, I., Molnár, T., Zajac, J., Belluš, J., Kekerová, S.: Simulácia robotického futbalu, tímový projekt. FIIT STU BA 2004.
http://www2.dcs.elf.stuba.sk/TeamProject/2004/team03/Download/DokumentaciaFinal.zip

2.2.5 CMUnited-99 (1998)

Základné informácie

Tím, vytvorený študentmi Carnegie Mellon University, dosiahol víťazstvo ligy Robocup v roku 1998. Tento návrh stavia na hráčovi z roku 1998 a snaží sa ho vylepšiť.

Návrh hráča

Návrh hráča sa odvíja od skutočnosti, že simulačný krok je napevno daný a má 100 ms. Správanie hráča je popísané pomocou jeho konania počas cyklu *t*:

- predpokladá sa, že hráč má obraz o stave sveta z cyklu *t-2* a vykonal akciu v čase *t-1*.
- pokiaľ je server v čase *t-1*, pri doručení vnemu (obraz, zvuk, dotyk) sa táto informácia ukladá v dočasných údajových štruktúrach. Obraz stavu sveta sa zatiaľ nemení.
- keď sa server ocitne v cykle *t* (hráč to zistí buď sledovaním hodín, alebo pri získaní vnemu s časovou pečiatkou *t*), hráč použije všetky doterajšie znalosti (dočasne uložený vnem a predikcia účinkov predošlých akcií), aby aktualizovať svoj stav tak, aby čo najviac korešpondoval so stavom serveru v čase *t-1*. Potom sa zvolí a serveru zašle akcia pre cyklus *t*.
- Nasleduje to isté pre cyklus *t+1*.

Na konci každého cyklu hráč určí, aký typ správania ma zaujať:

- snaha dať gól – tento záujem by mal mať vždy iba jeden hráč z celého tímu
- lokalizácia – ak hráč nevie, kde sa nachádza
- nájdenie lopty a udržanie optického kontaktu s ňou
- aktívna ofenzíva – čo najrýchlejšie sa priblíženie k lopte. Použité vtedy, ak žiaden iný spoluhráč nie je k lopte bližšie
- pomocná ofenzíva – ísť na pozíciu, kde je možná prihrávka. Použité vtedy, keď má iný spoluhráč loptu.
- aktívna defenzíva – pohyb k lopte, aj keď k nej už ide iný spoluhráč. Použité v defenzívnej časti ihriska.
- pomocná defenzíva – prenasledovanie protihráča.
- pasívna defenzíva – optické sledovanie protihráča, alebo presun na pozíciu, kde môže byť hráč užitočný.

Ďalšia časť návrhu je vytvorenie scenárov pri tzv. „dead-ball“ situáciách, akou je napríklad strela súperovho hráča na bránku. Cieľom je sériou prihrávok loptu dostať z vlastnej polovice ihriska.

Výhody návrhu

- typy správania sa hráčov
- preddefinované scenáre riešenia štandartných situácií
- návrh je úspešne overený a používa sa už niekoľko rokov.

Nevýhody návrhu

- architektúra hráča je prispôsobená pevným, 100 ms trvajúcim cyklom
- návrh nie je najnovší, je otázne, nakoľko je „zastaraný“

Zdroje

Stone, P., Veloso, M., Riley, P.: The CMUnited-99 Simulator Team 1999.
http://www.ep.liu.se/ea/cis/1999/007/02/cis9900702.pso.pdf

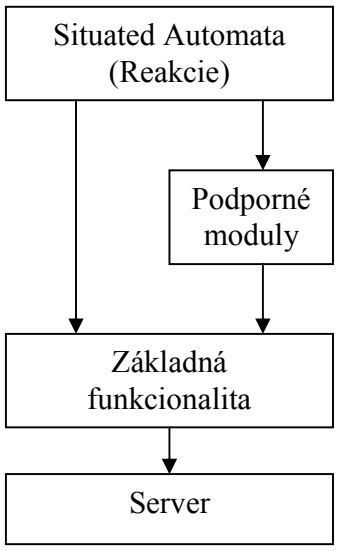
2.2.6 UBU (1999)

Základné informácie

Tento tím vytvorila The DECIDE Research Group, Department of Computer and Systems Sciences, Stockholm University. Bol vytvorený v roku 99, napísaný v jazyku Java a vychádzal z staršej verzie tímu od rovnakých autorov z roku 98.

Návrh hráča

Hráč je navrhnutý ako niekoľko nezávislých modulov, ktoré sú následne delené do vrstiev. Týmto zabezpečili jednoduché prepojenie tej istej inteligencie na iný typ simulácie (napr. malé roboty). Hlavné časti návrhu sú zobrazené na Obr. 2.1.



Obr. 6**.** Moduly a ich náväznosti.

Modul základnej funkcionality má za úlohu najmä preberanie informácií zo senzorov a tiež vykonávanie základných príkazov. Je rozdelená na 3 vrstvy:

- **Komunikačná** Má na starosti komunikáciu zo serverom, prípadne s inými hráčmi. Avšak komunikácia medzi hráčmi je v tejto verzií len okrajová.
- **Základné akcie** Vrstva základných akcií sprostredkováva vlastné rozhranie na vykonávanie jednotlivých akcií hráča. (kop do lopty, pohyb, a pod.)
- **Pamäťová** Pamäťová vrstva sa skladá z časti plánovacej a z krátkodobej pamäti.

Podporné moduly slúžia ako výpomoc a poskytujú informácie, ktoré server neposkytuje (napr. Ktorý tým má loptu, a pod.).

Modul zameraný na reakcie hráčov sa skladá z dvoch častí. Prvá sa zaoberá reaktívnym správaním (napr. odpískanie rozhodcom, a pod.) a druhá sa zaoberá premýšľaním hráča.

Výhody návrhu

Za výhody návrhu považujeme:

- Nápad podporných modulov
- Oddelenie platformovo závislých častí od zvyšku aplikácie
- Rozdelenie správania na reaktívne a premýšľanie

Nevýhody návrhu

Za najväčšiu nevýhodu tohto tímu považujeme implementáciu v jazyku Java. Toto však nijako nesúvisí s architektúrou, alebo inteligenciou hráčov. Nevýhody z hľadiska návrhu sú:

- Rozdelenie na moduly neobvyklým spôsobom
- Slabá, ba priam až žiadna komunikácia medzi hráčmi

Zdroje

Magnus Boman, Johan Kummeneje, David Lybäck, Håkan L. Younes: UBU Team
http://www.ep.liu.se/ea/cis/1999/007/30/cis9900730.pso.pdf

2.2.7 UvA (1999)

Základné informácie

UvA tím, tiež nazývaný aj Windmill Wanderers bol vytvorený na University of Amsterdam v roku 1999. Vychádza z predchádzajúcich verzií tímu od tých istých autorov. Je rozdelený na tri základné časti podľa funkcionality: Základná, Schopnosti, Ovládanie.

Návrh hráča

Hráč je rozdelený na tri základné vrstvy, z ktorých každá ma svoju špecifickú úlohu v systéme.

Základná vrstva sa snaží schovať komunikáciu so serverom pokiaľ možno čo najviac. Pri vytvorení hráča ktorý bude obsahovať len túto vrstvu (prípadne len minimum v ostatných) dostaneme Základného hráča.

Druhou vrstvou systému je vrstva schopností. Táto sa ďalej delí na:

- **Základné úlohy** Sem patria úlohy, ktoré sú samé o sebe jednoduché (napr. : nájdi loptu, kopni do lopty a pod.)
- **Hlavné úlohy** V tejto časti sú zahrnuté celistvé akcie ktoré môže hráč vykonávať (napr. preberanie lopty, strela, prihrávka a pod.)
- **Poskytovanie info** Táto časť má za úlohu získavanie a poskytovanie informácií ohľadne hry, rozostavenia hráčov, pozície lopty a pod.

Cieľom najvyššej, ovládacej, vrstvy je vybrať z množiny schopností tú najvhodnejšiu pre danú situáciu.

Obr. 7. Moduly hráča a ich náväznosti.

Tento hráč má tiež spracovanú zónovú stratégiu, ktorá má za úlohu priblížiť fungovanie tímu skutočnosti. Pretože hráč bez lopty sa snaží postaviť na takú pozíciu, kde bude v budúcnosti najužitočnejší. Túto vlastnosť sa pokúsili zapojiť aj autori tímu. Táto schopnosť má tiež za úlohu udržiavať správnu formáciu na ihrisku.
Každý hráč ma inak zadefinovaný priestor v ktorom sa môže pohybovať s loptou a bez lopty. Toto môže byť výhodná taktická pomôcka k dosiahnutiu dobrých výkonov.

Výhody návrhu

Za výhody opisovaného návrhu považujeme:

- Vhodne navrhnuté vrstvy a podvrstvy hráča
- Zónovú stratégiu tímu

Nevýhody návrhu

Za nevýhody tímu považujeme:

- Nezapojenie trénera do zónovej stratégie, ale tiež do rozhodovania hráča

Zdroje

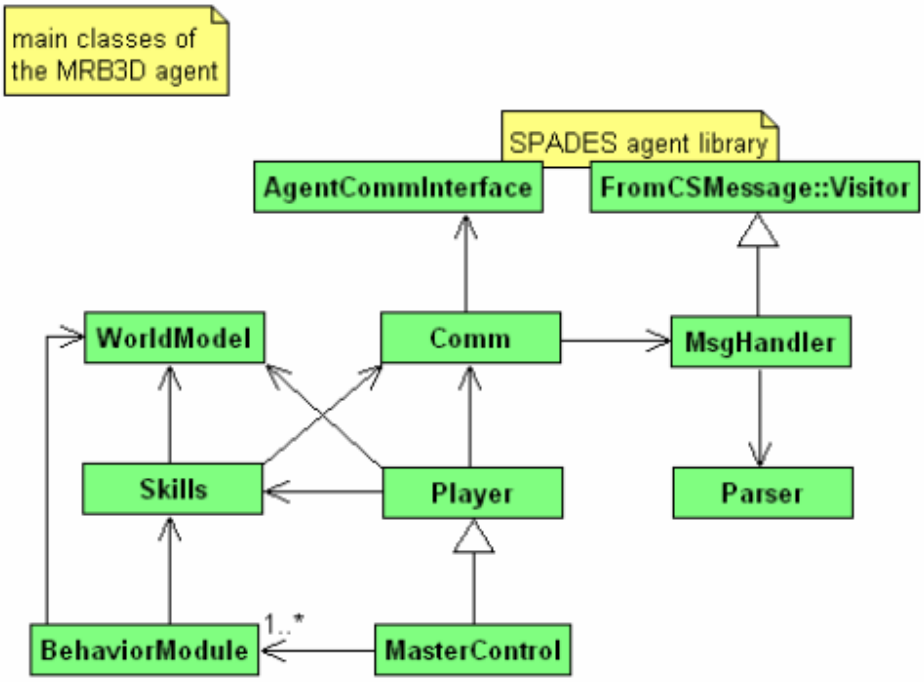
Emiel Corten, Frans Groen: UvA-Team.
http://www.ep.liu.se/ea/cis/1999/007/33/cis9900733.pso.pdf

2.2.8 Rolling Brains (2004)

Základné informácie

Tento návrh popisuje architektúru 3D tímu Rolling Brains z University Mainz z roku 2004. Návrh vychádza zo skúseností tejto výskumnej skupiny pri práci s 2D tímami. Tím sa zúčastnil majstrovstiev sveta Osaka 2005, ale do trojice najlepších tímov nepatril.

Návrh hráča



Obr. 8. Hlavné triedy 3D hráča.

Pre komunikáciu so serverom sa používa SPADES knižnica. Pre lokalizáciu agenta v priestore je použité pretínanie polyéderov.

Život agenta má nasledovné cykly:

- čakanie na správu od serveru a upravenie svojho modelu sveta na základe tejto správy
- na základe tejto predstavy agent vyberie akciu a pošle ju serveru
- čakanie na ďalšiu správu

Rozhodovacia vrstva agenta pozostáva z nasledovných častí:

- *BallHandling* – ovládania lopty a driblovanie s ňou
- *Pass modul* – prihrávanie lopty spoluhráčom
- *GoalShot* – modul pre strieľanie gólov

- *Positioning* – rozmiestnenie hráčov na ihrisku na základe stratégie (defenzívna/ofenzívna)
- *StandartSituation* – pre riešenie štandartných situácií

Každý z týchto modulov má procedúru *evaluate*, ktorá vracia diskrétnu hodnotu, označujúcu vhodnosť použitia tejto možnosti v danej situácii.

Výhody návrhu

- rozširovateľnosť návrhu
- robustný, ale flexibilný
- vychádza z fungujúceho a overeného návrhu 2D tímu
- možnosť zmeny na viacvláknový priebeh pri rozhodovaní sa agenta na základe príchodzej správy

Nevýhody návrhu

- tím sa s týmto návrhom nedostal medzi prvé tri tímy na súťaži v Osake 2005.

Zdroje

Arnold, A., Flentge, F., Schneider, C.: Team Description Mainz Rolling Brains 2004 - 3D
http://wombatz.informatik.uni-mainz.de/MRB2004_3D.pdf

2.2.9 Zhodnotenie analýzy tímov

Z analýzy architektúry už existujúcich hráčov vyplýva, že hráči úspešných tímov majú nasledovné črty;

- čo najväčší stupeň modulárnosti – jednotlivé časti sú vytvorené pomocou modulov s jednoduchým rozhraním. Toto umožní nahradenie akéhokoľvek modulu jeho novšou verziou.
- rozdelenie systému na vrstvy – každá vrstva má pevne stanovený význam(y). Doteraz sme identifikovali tieto funkcie: komunikáciu (server – hráč, hráč – hráč, hráč – tréner), plánovanie (vychádza z asynchrónnosti, viď. nasledujúcu odrážku), riadenie (tréner, vytváranie plánu, časovanie, stratégia, taktika), primitívy (pohyb, práca s loptou, ...)
- asynchrónnosť práce hráča – to, že čas na reakciu, ktorú má hráč vyhradenú serverom, je obmedzený (100 ms) neznamená, že vo zvyšnom čase nemôže pracovať na svojom „pláne“. Toto vytváranie, dopĺňanie a upravovanie plánu môže bežať nezávisle na tom, či je hráč v aktívnom stave (beží jeho 100 ms), alebo nie.

3 Špecifikácia požiadaviek

Naším cieľom je vytvoriť podklady pre jednoduchšiu tvorbu hráča 3D robotického futbalu. Najvhodnejším riešením je tvorba základných tried a jednoduchej funkcionality z hľadiska hry. Z výsledkov analýzy vyplýva, že našou úlohou je vytvoriť hráča prakticky od základov. Z toho možno odvodiť základné a nevyhnutné požiadavky na architektúru hráča, ktoré musia platiť súčasne. Architektúra by mala byť koncipovaná tak, aby tímy, ktoré nás budú nasledovať mali čo najmenšiu potrebu meniť ju v základoch.

Architektúra hráča teda musí byť:

- jednoduchá
  (z dôvodu, aby ďalšie tímy, ktoré budú pokračovať v našom diele, museli venovať čo najmenej času na pochopenie architektúry a mohli sa venovať ďalšiemu rozširovaniu)
- modulárna
  (s dôrazom na vysokú súdržnosť a nízku viazanosť častí, tak, aby mohlo možné nielen stavať nad existujúcim návrhom, ale meniť aj jeho základy, avšak aby tento systém motivoval k používaniu preddefinovanej štruktúry)
- rozširovateľná
  (aby ďalšie tímy mali čo najširšie pole pôsobnosti pri rozširovaní a vylepšovaní hráča)
- funkčná
  (je dôležité, aby už nami navrhnutý jednoduchý hráč bol schopný hry s minimálnou logikou)

Užitočným prvkom rovnako ako pre nás, tak aj do budúcnosti, je vytvorenie vlastného rozhrania na komunikáciu s hotovými časťami ako aj zadefinovaním štruktúry tried a modulov. Výhodné by bolo rozdeliť hráča na niekoľko častí, ktoré by boli na sebe len minimálne závislé a jednoducho zameniteľné.

3.1 Požiadavky na systém

Výsledný framework hráča bude vytvorený v jazyku C++ v prostredí Linux. Samotný produkt, vytváraný pomocou tohoto frameworku, bude musieť byť vytváraný vzhľadom na potreby serveru RoboCup-u 3D. Keďže tento server sa dynamicky mení, nie je možné dopredu odhadovať jeho náročnosť. Za cieľ si teda stanovujeme jej minimalizáciu, pričom budeme vychádzať zo súčasnej verzie servera (0.3), ktorá je efektívne spustiteľná v systéme. V súčasnej verzii serveru je pre samotnú činnosť hráča rezervovaných 0.100 s v každom ťahu, ktorý je vhodné maximálne využiť.

Hlavnou časťou by teda malo byť jadro hráča, o ktorého vývoj sa postaráme v prvom rade my. Toto by malo tvoriť rozhranie medzi už hotovými časťami a našim hráčom. Medzi jeho základné funkcie by sme mohli zaradiť:

- komunikácia so soccerserverom
  (samotné posielanie a prijímanie správ by malo byť zapuzdrené v nami vytvorených triedach tak, aby poskytovali čo najjednoduchšie komunikačné rozhranie, avšak aby bolo v prípade potreby možné ich jednoduché rozširovanie)
- úložisko informácií

(tu sa budú uchovávať najrôznejšie informácie o okolí – [model sveta hráča], ako aj ďalšie údaje, ktoré budú potrebné pre realizáciu zvolenej stratégie [model tímu]).
- riadenie vnútorných častí hráča
(niektoré kontakty z okolím môžu byť predspracované a poslané ďalším častiam hráča na spracovanie)

3.2 Vlastnosti a funkcionalita hráča

Ďalšími činnosťami by sa zaoberali vnútorné časti hráča. Funkcionalita tvorená vnútornými časťami hráča by sa dala rozdeliť na tri úrovne podľa jej zložitosti:

- primitívna funkcionalita
- stredná funkcionalita
- vyššia funkcionalita

Primitívna funkcionalita

- určenie polohy:
    1. vlastnej (samotného hráča)
    2. brány (či už vlastnej alebo súperovej)
    3. spoluhráčov (najbližších v okolí)
    4. protihráčov (najbližších v okolí)
    5. ostatných objektov na ihrisku (postranné čiary, hranice "šestnástky" a pod.)

- pohyb:
    1. bez lopty:
        1. k lopte (zachytenie lopty od súpera alebo prihrávky od spoluhráča)
        2. strategický presun:
            1. na konkrétnu pozíciu (pozícia vhodná na prihrávku pri stratégii)
            2. krytie protihráča
    2. s loptou – dribling (s cieľom dať gól alebo prihrať na gól)

- strela:
    1. na bránu (pokus hráča vsietiť gól)
    2. spoluhráčovi (prihrávka v rámci realizovanej stratégie)

Stredná funkcionalita

- tímová spolupráca na nižšej úrovni
(napr. komu momentálne prihrať v rámci realizovanej stratégie tímu a pod.)
- jednoduché stratégie
(z nich sa budú potom konštruovať stratégie komplexnejšie. Môže ísť napr. o stiahnutie hráčov do obranného pásma pri súperovom breaku a pod. Podľa charakteru sa delia na obranné, útočné a presunové)
- ohodnotenie situácie, v ktorej sa hráči nachádzajú
(či sa jedná sa o útočnú alebo obrannú akciu)
- reakcia na situáciu

(stanovenie zodpovednej jednoduchej stratégie, resp. postupnosti elementárnych činností na základe ohodnotenia situácie)

Vyššia funkcionalita

Zahŕňa prevažne zefektívnenie a doladenie funkcií strednej vrstvy vzhľadom na:

- tímovú spoluprácu a stratégie na vyššej (komplexnejšej) úrovni
  (stratégie ako postupnosť menších na seba nadväzujúcich stratégií)
- efektívnosť využívania energie hráča
  (aby hráč nerobil zbytočné pohyby navyše => voľba nasledovnej stratégie závisí od momentálneho stavu hráča a jeho okolitých spoluhráčov)
- učenie sa z predošlých udalostí
  (získavanie skúseností – úloha kouča [nebudeme implementovať])

Vzhľadom na našu východiskovú pozíciu a možnosti zatiaľ vyššiu funkcionalitu bližšie nešpecifikujeme, ale vytvoríme pre ňu podmienky pre naslednovníkov.

Dôležitým aspektom rozhodovania a myslenia hráča sú aj informácie o okolí, preto zaradíme do vnútornej časti hráča aj časti zaoberajúce sa zbieraním, spracovávaním a poskytovaním informácií o hre.

Medzi tieto časti radíme hlavne:

- model sveta hráča
  (hráč musí brať do úvahy, že informácie prichádzajúce cez receptory sú skreslené)
- model tímu, v ktorom sa hráč nachádza
  (teda v akých formáciach je tím schopný postupovať, aké taktiky podporuje a pod.)

Keďže začíname prácu na projekte s tématikou 3D robotického futbalu na FIIT ako prví a nemáme toľko hotových podkladov ako tímy z 2D RoboCupu, naším cieľom a hlavnou prioritou bude implementácia predovšetkým tých najjednoduchších činností hráča na úrovni primitívnej funkcionality a následne podľa časovej dispozícii aj časti činností z úrovne funkcionality strednej. Do konca zimného semestra plánujeme implementovať framework hráča, zahŕňajúci väčšinu jeho primitívnych funkcií typu Určenie polohy, Pohyb a Strela. V letnom semestri sa potom zameriame na implementovanie zvyšku týchto funkcií a pokúsime sa vytvoriť a aplikovať nejaké jednoduchšie stratégie a spôsoby myslenia hráčov v tíme.

4 Hrubý návrh

V tejto kapitole opisujeme hrubý návrh architektúry hráča. Vychádza z predchádzajúcich analýz tímov, pričom zohľadňuje požiadavky kladené v špecifikácií. Ide najmä o požadovanú flexibilitu architektúry a jednoduchosť z hľadiska pochopenia a používania.

V druhej časti sa zameriame na možnosti, ktoré zjednodušia tvorbu nových hráčov založených na našej architektúre.

4.1 Architektúra hráča

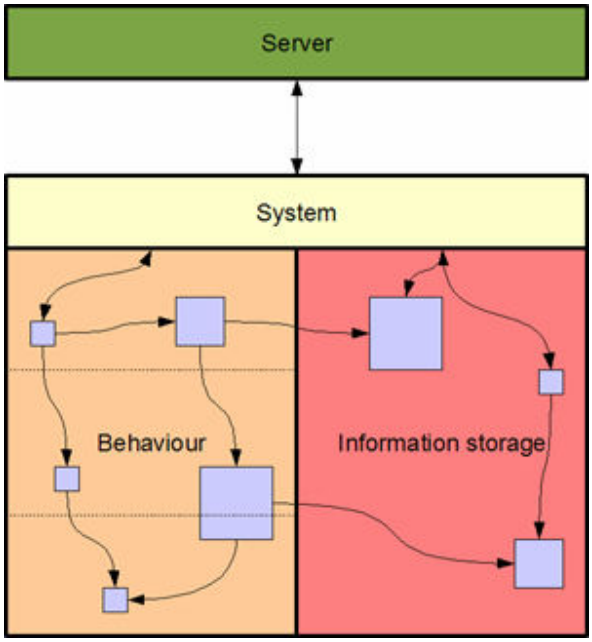
Väčšina hráčov, ktoré sme v predchádzajúcej časti analyzovali, obsahovala tieto časti:

- komunikačný modul – komunikácia so serverom
- modely sveta, tímu a hráča – informačné zdroje pre kognitívny modul
- kognitívny modul – samotná inteligencia hráča, rozhodovacie mechanizmy, centrum logiky hráča

Vychádzajúc z horeuvedených faktov zo zistení v analýze a zo špecifík 3D futbalu, vznikla architektúra, ktorá sa delí na tieto tri základné časti:

- systémová časť (systém)
- informačná časť (information storage)
- časť správania (behaviour)

Komunikácia medzi nimi je naznačená na nasledujúcom obrázku.



Obr. 9. Časti hráča a komunikácia medzi nimi. Na obrázku sú farebne odlíšené hlavné časti systému, v časti správania a v informačnej časti sú tiež zobrazené jednotlivé moduly danej časti. Šípkami je naznačený tok správ medzi modulmi.

Výhodou tejto architektúry oproti predchádzajúcej je zvýšenie granularity systému. Samotný hráč sa skladá z viacerých nezávislých modulov, ktoré sú voľne zviazané a ľahko vymeniteľné. Zvyšuje sa tak flexibilita voči prípadným zmenám servera či samotného hráča. Hráč dokonca nemá jasne definované vrstvy správania, pretože tento spôsob v minulosti viedol ku komplikáciám a neprehľadnému kódu.

Keďže v samotnom serveri sa nachádza knižnica zameraná na prácu s modulmi Zeitgeist, budeme ju využívať aj my. Jej hlavnou výhodou je dynamické pripájanie modulov do aplikácie, čo budú vyžadovať aj naše moduly. Táto knižnica využíva adresárovú štruktúru na udržanie stromu modulov.

Samotná štruktúra sa bude udržiavať aj preddefinovaním základných tried, z ktorých sa budú jednotlivé moduly vytvárať. Tieto budú špecifické pre každú časť a budú poskytovať informácie potrebné pre výkon funkcií daného modulu (napr. v IS module bude prístupné rozhranie na získavanie informácií zo senzorov, čo však nieje nutné v module správania).

4.2 Systémová časť

Jej hlavnou úlohou je zabezpečiť komunikáciu zo serverom. Poskytuje rozhranie na jednoduchšiu komunikáciu, získavanie dát alebo posielanie príkazov. Taktiež bude vykonávať volania smerom do vnútra hráča, napr. volanie vykonaj ťah a pod. Táto časť by sa mala meniť, len ak si to vyžiada nejaká vonkajšia zmena (napr. zmena servera). Musí byť však dostatočne flexibilná voči neskorším modifikáciám, keďže ďalší vývoj servera je viac ako pravdepodobný.

4.3 Informačná časť

Informačná časť obsahuje viacero modulov, ktoré majú za úlohu poskytovať hráčovi informácie o aktuálnom stave na ihrisku, ale tiež môžu predpovedať niektoré udalosti v najbližších cykloch (napr. pozícia lopty alebo hráča). Zaraďujeme sem modely hráča, tímu ako aj vnútorného sveta.

Všetky moduly budú poskytovať rovnaké rozhranie, pre základnú prácu s nimi. Toto bude realizované pomocou základnej triedy, z ktorej budú všetky tieto moduly odvodené. Ďalšie schopnosti jednotlivých modulov sú individuálne podľa ich použitia.

Samotné moduly by mali rozdeľovať celú informačnú časť na čo možno najmenšie časti, ktoré sú od seba len minimálne závislé. Jednotlivé moduly by mali čo najviac využívať okolité moduly a snažiť sa minimalizovať komunikáciu so serverom na potrebné minimum (napr. aby sa 3 moduly nepýtali čo hráč vidí).

4.4 Časť správania

Časť správania sa dá rozdeliť na tri vrstvy, ktoré sú však viac orientačného charakteru a mali by približne rozdeľovať moduly podľa toho, na akej úrovni vykonávajú akcie. V celej časti bude viacero modulov, ktoré zabezpečujú správanie samotného hráča. Snahou je zaistiť dostatočnú granularitu správaní, aby sa predišlo problémom s príliš malou flexibilitou. Samostatné moduly môžu byť napríklad modul bezpečného driblovania či modul rýchleho driblovania. Výhodou je, že takéto moduly je možné neskôr agregovať do ďalších modulov, ktoré budú rozhodovať z hľadiska globálnej stratégie a vyberať tak vhodný spôsob driblovania pre hráča v danej situácii.

Prepojením medzi časťou správania a systémom bude trieda, ktorá bude poskytovať rozhranie pre aktivovanie ťahu. Pomocou tohto rozhrania dostane časť správania správu, že má vymyslieť a vykonať ďalší ťah.

4.5 Tok riadenia v moduloch

Veľké množstvo modulov si vyžaduje vhodne navrhnuté riadenie, aby sa vykonávali všetky činnosti v správny čas.

Hlavné slovo v riadení bude mať systémová časť, ktorá bude rozhodovať čo sa udeje na základe prichádzajúcich správ. V prípade ak príde správa zodpovedajúca vykonaniu ťahu, systém zavolá najvyššiu triedu správania, ktorá bude rozhodovať o ďalších postupoch. V prípade ak príde správa o prijatí informácie z receptorov, táto bude predspracovaná a rozposlaná všetkým IS modulom, ktoré si prihlásili o jej rozposielanie. Tieto informácie spracujú a v prípade záujmu o nimi poskytované informácie ich poskytnú ďalej tým modulom, ktoré si ich vyžiadajú.

Ako príklad uvedieme príchod správy vykonaj ťah. Po príchode tejto správy dostane hlavná trieda správania správu, aby vykonala ťah. Táto môže vybrať kto a ako vykoná tento ťah. Ľubovoľný modul správania môže použiť údaje z informačných modulov. Ak tieto nemajú požadované údaje pošlú požiadavku na server a vyhodnotia odpoveď a na jej základe poskytnú

informácie. Postupným rozhodovaním na stále nižšej úrovni sa dostaneme až k vykonaniu samotných akcií hráčom a teda hráč vykoná ťah.

4.6 Podporné prostriedky pre programátorov

Prvotnou úlohou tohto projektu je zjednodušiť a zefektívniť vytváranie nových hráčov z pohľadu programátora. Náš systém nebude teda obsahovať len základnú sadu znovupoužiteľných modulov, ale aj rôzne makrá, ktoré obmedzia nutnosť písania opakujúceho sa kódu na úplné minimum. Pôjde najmä o makrá zjednodušujúce tvorbu nových modulov, pretože tak chceme podporiť programátorov k zvýšeniu granularity systému a vynútiť tak jednoduchosť a znovupoužiteľnosť jednotlivých modulov.

5 Zhodnotenie

Zhodnotenie doterajšej činnosti rozdelíme na dve časti. Prvou časťou bude hodnotenie tímovej spolupráce a druhou časťou bude hodnotenie predkladaného výsledku.

Náš tím prešiel obdobím formovania. Počas tohto obdobia sme si vyskúšali aké to je začať vytvárať tím zložený z ľudí, ktorý sa nepoznali. Fungovanie tímu v tomto období trpelo viacerými neduhmi, ktoré sa nám podarilo odstrániť zavedením pravidiel, ktorými sa budeme riadiť počas spolupráce na tomto projekte. Celkovo hodnotíme svoju spoluprácu kladne, pretože sa nám podarilo spojiť sily na dodanie požadovaného výsledku.

Predkladaný výsledok spĺňa všetky požiadavky kladené zo strany zadávateľa. Vypracovali sme analýzy existujúcich tímov z hľadiska ich architektúry. Na základe týchto analýz sme vypracovali špecifikáciu a hrubý návrh vznikajúceho systému.