Slovenská technická univerzita
Fakulta informatiky a informačných technológií

Tímový projekt II.

Inžinierske dielo

Tím #19 - WAWOT

Vedúci projektu: Ing. Ivan Kapustík
Predmet: Tímový projekt II.
Ročník: 2018/2019

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Projekt inteligentného robotického hráča simulovaného futbalu, ktorý je založený na najväčšej súťaži inteligentných autonómnych robotov RoboCup v sebe zlučuje vývoj softvéru umelej inteligencie, výskum robotiky a vývoj aplikácie v rámci fyzikálneho simulátora. Hráč, jeho pohyby, fyzika a jeho správanie je postavené na stanovených pravidlách 3D simulátora robotického futbalu. Jedná sa o pokračovanie v práci predchádzajúcich tímov, ktoré na vývoji a zdokonaľovaní hráča pracujú od roku 2010. Projekt je okrem predmetu Tímový projekt každoročne aj súčasťou zadaní bakalárskych a diplomových prác študentov našej fakulty. Okrem vytvorenia konkurencie schopného robotického hráča je jedným z hlavných cieľov tohto projektu rozvinúť schopnosti jednotlivých členov a naučiť ich pracovať v tíme. Nové úlohy sú identifikované na pravidelných stretnutiach, počas ktorých má každý možnosť vyjadriť svoj názor a obavy. Člen tímu si následne vyberie úlohy, ktoré chce riešiť a svoj progres zverejňuje na spoločnom komunikačnom kanáli - TFS, Slack a tak isto aj na spoločných tímových stretnutiach v štýle SCRUM. Synergiou dosiahnutou pri práci tímov ako aj jednotlivcov v rámci individuálnych projektov vzniká už 8. rok znalostná databáza s množstvom akumulovaného know-how, ktoré je využiteľné pri ďalšom vývoji.

Našou úlohou v tomto semestri bolo predovšetkým využiť doposiaľ nadobudnuté informácie od nás samotných ako aj od predchádzajúcich tímov a implementovať začaté a navrhované zmeny v projekte. Tieto zmeny sme dôkladne zdokumentovali na wiki projektu a nové implementácie otestovali s dôrazom na zachovanie modularity a integrity projektu. Výsledkom práce je aktuálne zrýchlené rozhodovanie hráča v špecifických situáciách zahŕňajúcich najmä rozhodovanie pri slabej orientácií na ploche - dôsledok implementácie rozlišovania vzorov čiar na ihrisku, odstránenie mnohých chybových stavov v behu ako aj prehľadnejšie návody na inštaláciu a prehľadnejšia wiki. Jednou z veľkých výziev projektu bolo tiež odstránenie nepoužívaných a prázdnych tried, metód a importov. Po 10 ročnom vývoji, na ktorom sa vystriedali desiatky študentov, bolo potrebné projekt výrazne sprehľadniť a aktualizovať aby bola akákoľvek ďalšia práca na ňom možná a efektívna.

Výsledok našej práce, ako dúfame, umožní značne ľahší štart vývoja nasledujúcemu tímu. Zostávajúcu prácu na projekte sme analyzovali, kategorizovali a uverejnili pre ďalšie tímy na wiki.

1. Manažérske roly

Bc. Adriana Beňovičová, Bc. Zoltán Csengődy, Bc. Ľubomír Fischer,

Bc. Andrej Kútny, Bc. Christoper Liebe, Bc. Adam Mikolášek, Bc. Kitti Nagyová

V tejto časti dokumentu sú opísané manažérske roly, ktoré sme identifikovali a priradili jednotlivým členom tímu.

1.1. Identifikované roly

V rámci riešenia tímového projektu sme identifikovali nasledovné roly: Hlavný projektový manažér, manažér dokumentovania, manažér špecifikácie úloh, manažér verziovania, manažér testovania, manažér komunikácie a správy dát, manažér kvality a manažér rizík.

1.2. Opis rolí

Táto kapitola obsahuje opis jednotlivých rolí a ich úlohy.

1.2.1. Hlavný projektový manažér

Bc. Filip Dian

Hlavný manažér tímu by mal viesť tím, motivovať k efektívnemu plneniu úloh jeho členov a v neposlednom rade dbať o celkový vzhľad fungovania tímu. Má takisto rolu Scrum Mastera a teda sa stará o riadenie tímovej práce v rámci nástrojov, ktoré si tím na kolaboráciu zvolil a snaží sa čo najviac skvalitniť prácu s nimi. Riadi priebeh stretnutí a ich obsah. Popri vedení tímu takisto pracuje na úlohách súvisiacich so spravovaním šprintov a vývoja ako sú súhrnné dokumentácie šprintov a kontroluje vytvárané úlohy a ich plnenie. Jeho úlohy sú v krátkosti teda:

- riadenie tímovej práce v rámci kolaboračných nástrojov
- motivovanie členov k plneniu úloh
- riadenie stretnutí
- skvalitnenie pracovného prostredia a zjednodušenie riadenia vývoja
- spracovávanie dokumentácie týkajúcej sa šprintov
- kontrola úloh tímu a ich plnenie
- kontrola práce ostatných rolí v tíme

1.2.2. Manažér dokumentovania

Bc. Andrej Kútny

Manažér dokumentovania je zodpovedný za správne dokumentovanie úloh, ktoré tím ukončí. Na základe dokumentácie musí byť jasné, na čom daný člen pracoval a aký je záver jeho práce. Manažér dohliada na konzistentnosť a jasnosť týchto dokumentov. Preto je potrebné aby manažér stanovil základnú štruktúru dokumentov a štylistiku písania. Taktiež je potrebné, aby všetky zverejnené dokumenty manažér skontroloval a vyhol sa tým prípadným nedostatkom formálnej stránky dokumentov. V prípade porušenia predom stanovených podmienok môže vrátiť autorovi dokument s konkrétnymi pripomienkami na nápravu. Základné úlohy manažéra dokumentácie teda sú:

- vytvorenie vzorových dokumentov a dokument popisujúci štylistiku písania
- reviduje dokumentáciu vytvorenú ostatnými členmi
- pri nesprávne napísanom dokumente dokument vráti autorovi s pripomienkami na opravu
- reviduje zápisnice a presúva ich do To publish

1.2.3. Manažér špecifikácie úloh

Bc. Ľubomír Fischer

Manažér špecifikácie úloh sa stará o formálnu a obsahovú špecifikáciu úloh v rámci projektu. Na začiatku šprintu je nevyhnutné čo možno najpresnejšie pochopenie obsahu jednotlivých úloh naplánovaných v ďalšom šprinte pre predchádzanie chybám spojeným s nesprávnym, či neúplným vypracovaním zadanej úlohy. Správne pochopenie zadania úlohy je závislé od vzájomnej komunikácie manažéra špecifikácie úloh s osobou, ktorá danú úlohu vytvorila a ktorá ju dostala priradenú. Jeho hlavnými úlohami v rámci práce a plánovania v tíme teda sú:

- Presné definovania zadania konkrétnej úlohy na nižšej aj vyššej vrstve abstrakcie
- Otvorenie komunikácie medzi zadávateľom a riešiteľom úlohy pre správne pochopenie zadania
- Spísanie formálnej špecifikácie úloh a ich akceptačných kritérií

1.2.4. Manažér verziovania

Bc. Adam Mikolášek

Manažér verziovania má na starosť správne využívanie git úložiska projektu. V tomto projekte je využité úložisko BitBucket. Manažér verziovania je zodpovedný za správne využívanie repozitáru ostatnými členmi tímu a za udržiavanie prehľadného repozitáru. Jeho hlavnými úlohami teda sú:

- Vytvára metodiky písania a commitovania kódu
- Stará sa o chod BitBucket repozitáru
- Kontroluje vytvárania a mergeovania vetiev

1.2.5. Manažér testovania

Bc. Kitti Nagyová

Manažér testovania má na starosť overovanie funkčnosti softvéru počas celej doby vývoja. Manažér testovania vytvára a konzultuje testovacie scenáre s ostatnými členmi tímu pre jednotlivé úlohy. Stará sa o to, aby každá jedna úloha bola otestovaná. V prípade zmien implementácie testovacieho frameworku spravuje JUnit testy.

Úlohy manažéra testovania:

- Plánuje testovanie
- Vytvára testovacie scenáre a vyhodnocuje testy
- Spravuje unit testy

1.2.6. Manažér komunikácie a správy dát

Bc. Zoltán Csengődy

Manažér komunikácie a správy dát úzko spolupracuje s členmi tímu vrátane s vlastníkom softvérového produktu pre zabezpečenie plynulej komunikácie v rámci tímu a odstránenie vzniknutých nejasností vyplývajúcich zo slabej komunikácie medzi členmi tímu. Hlavnou úlohou manažéra komunikácie a správy dát je identifikácia a spojazdnenie vhodných komunikačných kanálov a nástrojov pre plynulú komunikáciu medzi členmi tímu. Má za úlohu vytvoriť spoločnú e-mailovú schránku (Google Groups), ku ktorej majú mať prístup všetci členovia tímu vrátane vlastníka daného softvérového produktu. Zodpovedá za správu zvoleného skupinového komunikačného nástroja (Slack). Správa takéhoto nástroja zahŕňa správne nastavenie nástroja, manažovanie jeho komunikačných kanálov a určenie pravidiel komunikácie. Jeho ďalšou dôležitou úlohou je určenie spôsobu uchovávania a zálohovania dát a dokumentov vzniknutých počas vývoja softvérového produktu. Táto úloha zahŕňa zvolenie

spoľahlivého nástroja na uchovávanie a správu dát (Team Foundation Server) a vypracovanie stratégie obnovy dát v prípade ich straty.

Úlohy manažéra komunikácie a správy dát:

- Vytvorenie metodiky komunikácie
- Správa skupinového komunikačného nástroja
- Správa spoločnej e-mailovej schránky
- Správa uchovávania a zálohovania dát

1.2.7. Manažér kvality a obsahu na webe

Bc. Christopher Liebe

Správca obsahu je zodpovedný za vytvorenie webovej prezentácie tímu, publikovanie nových dokumentov a aktualizáciu vedomostnej databázy (wiki). V prípade potreby implementuje novú funkcionalitu a odstraňuje chyby.

Zároveň má za úlohu sledovať výstup ostatných členov tímu, kontrolovať či sú dodržiavané stanovené konvencie a poukázať na prípadné nedostatky. Medzi jeho kompetencie patrí:

- Priebežne prezerá commit-y v repozitári pred schválením pull requestu, pričom sa zameriava na dodržanie konvencie písanie kódu
- Sleduje funkčnosť aplikácie
- Uplatnenie metodík pri tvorbe softvérového produktu

1.2.8. Manažér rizík

Bc. Adriana Beňovičová

Náplňou manažéra rizík je stanoviť a zadefinovať možné riziká, ktoré počas našej práce na tímovom projekte môžu nastať. Je dôležité vopred a korektne vytvoriť metodiku rizík. Súčasne je potrebné neustále sledovať, aké ďalšie nešpecifikované riziká môžu nastať a snažiť sa im predísť. V nepriaznivej situácii, ak riziko už nastane, je dôležité, aby manažér rizík zaujal hlavnú rolu a našiel čo najoptimálnejšie riešenie s najmenšími škodami či časovými sklzmi.

Hlavné body náplne práce manažéra rizík sú:

- Vytvoriť metodiku rizík
- Neustále analyzovanie nových možných nezadefinovaných rizík
- Čo najoptimálnejšie riešiť už vzniknuté problémy a efektívne ich odstrániť

1.3. Rozdelenie manažérskych úloh

Bc. Kitti Nagyová

Počas prvých deviatich týždňov semestra boli úlohy v tíme rozdelené nasledovne, ako zobrazuje **Error! Reference source not found.**.

Hlavný projektový manažér	Bc. Filip Dian
Manažér dokumentovania	Bc. Andrej Kútny
Manažér špecifikácie úloh	Bc. Ľubomír Fischer
Manažér verziovania	Bc. Adam Mikolášek
Manažér testovania	Bc. Kitti Nagyová
Manažér komunikácie a správy dát	Bc. Zoltán Csengődy
Manažér kvality	Bc. Christopher Liebe
Manažér rizík	Bc. Adriana Beňovičová

Tabuľka 1 Rozdelenie úloh v tíme

1.4. Aplikácie manažmentov

V priebehu akademického roka sme zaviedli viacero postupov, ktorými sa pri tvorbe softvéru a jeho testovaní riadime. Sú súčasťou prílohy. Vytvorili sme nasledovné metodiky a postupy:

- Metodika komunikácie
- Metodika testovania
- Metodika písania kódu
- Metodika commitovania kódu
- Metodika mergovania
- Definition of Ready
- Definition of Done

2. Sumarizácie šprintov

2.1. Šprint č. 1

2.1.1. Základné informácie

Začiatok: 9.10.2018

Koniec: 23.10.2018

2.1.2. Retrospektíva

Dňa 23.10. 2018 sme ukončili šprint č. 1. Po jeho uzavretí sme urobili retrospektívu, počas ktorej sa každý člen vyjadril, čo sa mu páčilo a nepáčilo a určili si body, ktoré sa budeme v budúcnosti snažiť splniť. Pri bodoch sme aj určili, kto sa postará o jeho splnenie a v niektorých prípadoch aj vytvorili task.

2.1.2.1. Čo sa podarilo:

- veľmi dobrá spolupráca a komunikácia (Filip, Adam)
- nastavenie nástrojov pre budúce šprinty (Filip, Ľubomír)
- dokončená väčšina úloh v šprinte (Filip)
- priebežná práca na taskoch (Ľubomír)
- spokojný so všetkým (Chris, Adriana, Zoltán)
- všetko sa nakoniec vyriešilo (Chris)
- využívanie Slacku (Andrej)
- dochádzka na stretnutia (Andrej)
- nastavenie TFS (Andrej, Kitti)
- dobre odhadnutie práce do šprintu (Adam)
- zohranie tímu (Adriana)

2.1.2.2. Čo by sa mohlo zlepšiť:

- využívanie TFS (Filip)
- aktualizovanie stavu taskov (Filip)

- skupinová špecifikácia taskov (Ľubomír, Adriana)
- sledovanie stavu úloh (Ľubomír)
- nejasné názvoslovie dokumentov (Andrej)
- dokumenty v PDF sa nedajú upravovať (Andrej)
- informovanosť o stave taskov - review, testing, done.. (Kitti)
- chýba Definition of ready (Adam)
- žiadne rozdelenie review (Adam, Zoltán)
- spomalené dokončovanie úloh kvôli závislostiam (Zoltán)

2.1.2.3. Akčné body:

- priebežné aktualizovanie TFS (všetci)
- jasnejšia špecifikácia taskov (solution owner)
- sledovanie Slacku a TFS (všetci)
- krátky stand-up na každom stretnutí (scrum master + všetci)
- pridávanie ID a názvu tasku do názvu dokumentov (všetci)
- do TFS upload dokumentov vo formáte .docx, na stránku .pdf (všetci)
- oznámenie o presunutí tasku do review (všetci)
- integrácia TFS so slackom na notifikácie (Filip)
- vytvorenie Definition of ready (Chris)
- určenie, kto bude robiť review, najlepšie pri vytváraní taskov (všetci)
- ustrážiť závislosti medzi taskami (scrum master)
- viac dbať na dokončenie podľa definície (všetci)

2.1.3. Práca tímu

Na začiatku šprintu sme identifikovali prácu, ktorá by mala byť dokončená v rámci šprintu č. 1. Na základe vytvorených user stories sme zadali jednotlivé úlohy pre ich splnenie. Väčšina úloh bola vytvorená na začiatku šprintu a niektoré sa doplnili neskôr.

2.1.3.1. User stories

Do šprintu boli zaradené nasledujúce user stories.

ID	Title	Effort	State
8951	Vytvorenie webstránky	5	Done
8954	Zápisy	1	Done
8958	Optimalizácia kódu	8	Done
8961	Porovnanie iných implementácií	3	Done
9065	Analýza orientácie hráča v prostredí	8	Done
9186	Opis inštalácie	2	Done
9191	Príprava tímovej práce	2	Done

Tabuľka 2 User stories v šprinte č. 1

2.1.3.1.1. Opis user stories

- 8951 – Vytvorenie webstránky – inštalácia webového servera a vytvorenie základu webovej stránky tímu
- 8954 – Zápisy – spracovanie zápisníc zo stretnutí
- 8958 – Optimalizácia kódu – analýza výpočtovej náročnosti riešenia a implementácia vylepšení pre jej zníženie
- 8961 – Porovnanie iných implementácií – analýzy riešení iných tímov, ich fungovanie jednotlivých častí a spôsobu komunikácie
- 9065 – Analýza orientácie hráča v prostredí – analýza momentálneho riešenia problematiky orientácie hráča v rámci ihriska
- 9186 – Opis inštalácie – spracovanie postupov inštalácie prostredí na jednotlivé systémy pre zverejnenie na Wiki

- 9191 – Príprava tímovej práce – vytvorenie vzorových dokumentov a nastavenie nástrojov pre podporu ďalšej tímovej práce

2.1.3.2. Úlohy

Počas šprintu boli identifikované nasledujúce úlohy.

ID	Title	Assigned To	Priority	State	Completed Work
8950	Pridať všetkých členov do TFS	Bc. Filip Dian	1	Done	0,5
8952	Rozbehanie hostingu	Bc. Christopher Liebe	1	Done	3
8953	Vytvorenie webstránky tímu	Bc. Christopher Liebe	1	Done	4
8955	Upraviť zápisnice podľa vzoru	Bc. Andrej Kutny	2	Done	0,5
9066	Yourkit licencie	Bc. Filip Dian	1	Done	0,5
9067	Analýza java kódu (yourkit)	Bc. Filip Dian	1	Done	6
9068	Nájdenie ostatných zdrojových kódov a dokumentácií	Bc. Lubomir Fischer	1	Done	3
9069	UDP/TCP komunikácia so serverom u iných hráčov	Bc. Lubomir Fischer	2	Done	2
9181	Analýza súčasného riešenia	Bc. Adriana Benovicova	1	Done	4,5
9182	Analýza matíc u iných hráčov	Bc. Kitti Nagyova	1	Done	5
9183	Prejdenie logiky euklidovských matíc	Bc. Kitti Nagyova	1	Done	3
9184	Spracovanie zápisnice č.3	Bc. Andrej Kutny	2	Done	0,5
9185	Spracovanie zápisnice č.4	Bc. Andrej Kutny	2	Done	0,5
9187	Inštalácia na Windows	Bc. Adam Mikolasek	1	Done	2,5
9188	Inštalácia na Mac	Bc. Zoltan Csengody	1	Done	3

9189	Aktivovanie Wiki	Bc. Christopher Liebe	2	Done	1
9190	Vytvorenie dokumentu Definition of done	Bc. Filip Dian	1	Done	1
9197	Spracovanie na wiki	Bc. Christopher Liebe	2	Done	1
9280	Vytvorenie vzorových dokumentov	Bc. Kitti Nagyova	1	Done	1
9281	Build a analýza videnia tímu Austin Villa	Bc. Andrej Kutny	2	Done	4
9282	Nastavenie TFS	Bc. Filip Dian	1	Done	1
9380	Spracovať doterajšie zápisnice podľa nového vzoru	Bc. Andrej Kutny	2	Done	0,5

Tabuľka 3 Úlohy v šprinte č. 1

2.1.3.2.1. Opis úloh

- 8950 – Pridať všetkých členov do TFS – pridanie všetkých členov tímu do nástroja TFS na serveri tfs.fiit.stuba.sk
- 8952 – Rozbehanie hostingu – počiatočná inštalácia webového servera tímovej stránky
- 8953 – Vytvorenie webstránky tímu – vytvorenie základu webovej stránky tímu
- 8955 – Upraviť zápisnice podľa vzoru – úprava doterajších napísaných zápisníc do vzoru, ktorý sa bude používať počas celého projektu
- 9066 – Yourkit licencie – zabezpečenie licencií pre program Yourkit Java Profiler na analyzovanie rôznych aspektov fungovania Java kódu
- 9067 – Analýza Java kódu (Yourkit) – analyzovanie výpočtovej a pamäťovej náročnosti riešenia
- 9068 – Nájdenie ostatných zdrojových kódov a dokumentácií – vyhľadanie informácií (zdrojové kódy, dokumentácia..) o fungovaní hráčov iných zahraničných tímov
- 9069 – UDP/TCP komunikácia so serverom u iných hráčov – zistenie, ktorý protokol je používaný inými tímami a porovnanie, čo je lepšie

- 9181 – Analýza súčasného riešenia – zistenie, aké techniky jednotlivých častí fungovania hráča sa momentálne používajú v projekte
- 9182 – Analýza matíc u iných hráčov – analýza použitia euklidovských matíc v riešeniach iných tímov
- 9183 – Prejdenie logiky euklidovských matíc – analýza logiky euklidovských matíc pre budúce implementovanie
- 9184 – Spracovanie zápisnice č.3 – vyhotovenie zápisnice z 3. stretnutia
- 9185 – Spracovanie zápisnice č.4 – vyhotovenie zápisnice z 4. stretnutia
- 9187 – Inštalácia na Windows – spracovanie postupu inštalácie prostredia na systém Windows
- 9188 – Inštalácia na Mac – spracovanie postupu inštalácie prostredia na systém Mac
- 9189 – Aktivovanie Wiki – rozbehnutie Wiki na webovej stránke tímu
- 9190 – Vytvorenie dokumentu Definition of done – určenie Definition of done pre user stories a spracovanie do dokumentu
- 9197 – Spracovanie na wiki – upravenie dokumentu Definiton of done pre upload na web stránku
- 9280 – Vytvorenie vzorových dokumentov – spracovanie vzorov pre dokumenty, ktoré sa budú písať počas šprintu
- 9281 – Build a analýza videnia tímu Austin Villa – analýza spôsobu, akým sa vyhodnocuje videnie hráča zahraničného tímu Austin Villa
- 9282 – Nastavenie TFS – počiatočné nastavenie nástroja TFS pre tímovú prácu
- 9380 – Spracovať doterajšie zápisnice podľa nového vzoru – upravenie nových zápisníc podľa nového vzoru

2.1.4. Zhodnotenie

V šprinte sa nám podarilo dokončiť všetku zadanú prácu, na základe ktorej sme mohli aj vytvoriť user stories do ďalšieho šprintu. Išlo hlavne o rozbehanie prostredí, webstránky a

analýzu súčasného stavu v našom, ale aj iných riešeniach, aby sme vedeli identifikovať prácu do nasledujúcich šprintov.

2.1.4.1. Podiel práce členov tímu

Podľa vykazovania práce na taskoch bola vytvorená tabuľka podielov práce jednotlivých členov tímu.

Člen tímu	Formálne stretnutia	Neformálne stretnutia	Práca na taskoch
Bc. Filip Dian	6h	8h	9h
Bc. Christopher Liebe	6h	8h	9h
Bc. Andrej Kutny	6h	8h	6h
Bc. Lubomir Fischer	3h	8h	5h
Bc. Adriana Beňovičová	6h	8h	4,5h
Bc. Kitti Nagyova	6h	8h	9h
Bc. Adam Mikolasek	6h	8h	2,5h
Bc. Zoltan Csengody	3h	8h	3h

Tabuľka 4 Podiel práce v 1. šprinte

2.2. Šprint č. 2

2.2.1. Základné informácie

Začiatok: 24.10.2018

Koniec: 13.11.2018

2.2.2. Retrospektíva

Dňa 13.10. 2018 sme ukončili šprint č. 2. Po jeho uzavretí sme urobili retrospektívu, počas ktorej sa každý člen vyjadril, čo sa mu páčilo a nepáčilo a určili si body, ktoré sa budeme v budúcnosti snažiť splniť. Pri bodoch sme aj určili, kto sa postará o jeho splnenie a v niektorých prípadoch aj vytvorili task.

2.2.2.1. Čo sa podarilo:

- väčšia miera korektnosti a používania TFS (Filip, Adam, Kitti)
- Ľubomír rýchlo prebral nedokončenú prácu Chrisa (Filip)
- výborná komunikácia (Filip, Zoltán)
- reálny progres v zlepšovaní programu (Ľubomír)
- dokončenie formálnej stránky práce tj. metodiky, komunikácia, kolaboračné nástroje apod. (Ľubomír)
- Chrisova snaha s webom a optimalizáciou komunikácie pomocou threadov (Ľubomír)
- lepšia kolaborácia vo vývoji (Adriana)
- prepojenie TFS so Slackom (Adam, Zoltán)
- systém zadávania rolí v rámci taskov (Kitti)
- Ľubomírova revízia analýzy videnia hráča Austin Villa (Andrej)
- práca Scrum mastera (Andrej)
- dokončenie všetkých taskov (Andrej, Zoltán)
- poctivé dokončenie konca šprintu (Andrej)

2.2.2.2. Čo by sa mohlo zlepšiť:

- dokončovanie taskov posledné dni šprintu (Filip, Ľubomír, Adam, Kitti, Andrej, Zoltán)
- slabá kontrola v revíziách, aj po obsahovej aj po formálnej stránke (Filip)
- veľa recyklovania v rámci analýz, malá snaha o nový obsah (Ľubomír)
- málo komunikácie ohľadom špecifikácie user stories, stále dochádza k nedorozumeniam ohľadom obsahu (Ľubomír, Adam)

- neskoré revízie taskov (Adriana)
- neskoré publikovanie dokumentov na stránku (Kitti)
- slabá efektívnosť spoločných stretnutí (Andrej)
- málo publikovaných analýz na wiki, moli by v budúcnosti chýbať (Andrej)
- zlý stav wiki tj. obsah, dizajn (Andrej)
- nesledovanie vytvorených metodík (Andrej)
- stále malý prehľad o témach a fungovaní projektu (Zoltán)

2.2.2.3. Akčné body:

- snažiť sa dokončovať tasky priebežne, začať pracovať na taskoch v prvej polovici šprintu (všetci)
- prísnejšie zadávanie pripomienok po revíziach a kontrola po následnej oprave (všetci)
- viac sa snažiť o prínos práce do projektu (všetci)
- viac diskutovať o špecifikácii s cieľom korektne ju pochopiť (Solution owner + všetci)
- skoršia revízia (všetci)
- publikovanie dokumentov na stránku najneskôr do konca dňa stretnutia (Web master)
- snažiť sa viac pracovať na spoločných stretnutiach (všetci)
- publikovanie všetkých analýz a kontrola, ktoré chýbajú (Web master)
- prekopať štruktúru wiki a dizajn (Web master)
- viac si prejsť metodiky + kontrola obsahu aj úpravy (všetci)
- prejsť si doterajšie analýzy a obsah wiki (všetci)

2.2.3. Práca tímu

Na začiatku šprintu sme identifikovali prácu, ktorá by mala byť dokončená v rámci šprintu č. 1. Na základe vytvorených user stories sme zadali jednotlivé úlohy pre ich splnenie. Väčšina úloh bola vytvorená na začiatku šprintu a niektoré sa doplnili neskôr.

2.2.3.1. User stories

Do šprintu boli zaradené nasledujúce user stories.

ID	Title	Effort	State
8957	Analýza orientácie hráča	5	Done
8962	Zero moment point	3	Done
9559	Publikovanie dokumentov na wiki	1	Done

9563	Zdokumentovanie metodiky	3	Done
9565	Implementácia vylepšení spracovania komunikácie	8	Done
9568	Nastavenie TFS	2	Done
9572	Zápisnice	1	Done
9586	Špecifikácie user story	3	Done
9644	Optimalizácia kódu	5	Done

Tabuľka 5 User stories v šprinte č. 2

2.2.3.1.1. Opis user stories:

- 8957 – Analýza orientácie hráča – analýza súčasného riešenia zameraná na spôsob orientácie hráča vo virtuálnom prostredí
- 8962 – Zero moment point – opis konceptu ZMP a analýza súčasného použitia v programe a možnosti jeho zavedenia
- 9559 – Publikovanie dokumentov na wiki – spracovanie doterajších dokumentov a nahratie na wiki
- 9563 – Zdokumentovanie metodiky – spracovanie metodík tímovej práce, ktoré budú dodržované nasledujúce šprinty
- 9565 – Implementácia vylepšení spracovania komunikácie – optimalizácia identifikovaných problémových častí kódu, ktoré sú zbytočne výpočtovo náročné
- 9568 – Nastavenie TFS – pridanie nových atribútov pre lepšiu informovanosť o práci na taskoch a integrácia so Slackom pre notifikácie o ich stave
- 9572 – Zápisnice – priebežné spracovanie zápisníc
- 9568 – Špecifikácie user stories – vytvorenie špecifikácií pre pridané user stories
- 9644 – Optimalizácia kódu – identifikovanie problémových častí a vytvorenie taskov na budúci šprint

2.2.3.2. Úlohy

Počas šprintu boli identifikované nasledujúce úlohy.

ID	Title	Assigned To	Priority	State	Completed Work
9560	Spracovanie inštalácie prostredí	Bc. Christopher Liebe	2	Done	1

9564	Vytvorenie Definition of ready	Bc. Christopher Liebe	2	Done	1
9566	Pridanie viac vlákien do spracovania komunikácie	Bc. Christopher Liebe	2	Done	1,5
9567	Odstránenie posielania správ a výpisov do logu	Bc. Filip Dian	2	Done	3
9569	Integrácia so Slackom	Bc. Filip Dian	2	Done	0,5
9570	Pridanie prechodu medzi In review a Done	Bc. Filip Dian	2	Done	0,5
9571	Analýza aktuálneho stavu	Bc. Andrej Kútny	2	Done	2
9573	Spracovanie zápisnice č. 6.0	Bc. Kitti Nagyová	2	Done	1
9574	Spracovanie zápisnice č. 6.1	Bc. Andrej Kútny	2	Done	0,5
9575	Spracovanie zápisnice č. 6.2	Bc. Andrej Kútny	2	Done	0,5
9576	Spracovanie zápisnice č. 7.0	Bc. Kitti Nagyová	2	Done	1
9577	Spracovanie zápisnice č. 7.1	Bc. Andrej Kútny	2	Done	0,5
9578	Spracovanie zápisnice č. 7.2	Bc. Andrej Kútny	2	Done	0,5
9579	Analýza zisťovania polohy a orientácie hráča	Bc. Adriana Beňovičová	2	Done	5
9580	Metodika písania a commitovania kódu	Bc. Adam Mikolášek	2	Done	2
9581	Metodika testovania tasku	Bc. Kitti Nagyová	2	Done	2
9582	Metodika komunikácie	Bc. Zoltán Csengődy	2	Done	4
9583	Metodika review kódu	Bc. Adriana Beňovičová	2	Done	4

9584	Metodika pracovania s TFS	Bc. Filip Dian	2	Done	2
9585	Export taskov pre šprint č. 1	Bc. Filip Dian	2	Done	0,5
9587	Spracovanie špecifikácie user story šprintu č. 2	Bc. Ľubomír Fischer	2	Done	6
9588	Spracovanie metodík	Bc. Andrej Kútny	2	Done	0,5
9639	Spracovanie retrospektívy z 1. šprintu	Bc. Filip Dian	2	Done	1
9640	Pridanie atribútov na task	Bc. Filip Dian	2	Done	0,5
9641	Nastavenie notifikovania Slacku podľa tagu	Bc. Filip Dian	2	Done	0,5
9643	Analýza kalmanovho filtra hráča	Bc. Adam Mikolášek	2	Done	5
9645	Vytvorenie Backlog Taskov na základe pripomienok a analýzy v yourkit	Bc. Kitti Nagyová	2	Done	2

Tabuľka 6 Úlohy v šprinte č. 2

2.2.3.2.1. Opis úloh

- 9560 – Spracovanie inštalácie prostredí – úprava návodov na inštaláciu prostredia a nahratie na stránku
- 9564 – Vytvorenie Definition of ready – spracovanie Definition of ready pre user stories
- 9566 – Pridanie viac vlákien do spracovania komunikácie – optimalizácia spracovania komunikácie, ktorá bežala iba na jednom vlákne
- 9567 – Odstránenie posielania správ a výpisov do logu – vypnutie odosielania správ do TestFrameworku a vypisovania do logu keď nie je potrebné
- 9569 – Integrácia so Slackom – integrovanie TFS so službou Slack pre automatické odosielanie notifikácií
- 9570 – Pridanie prechodu medzi In review a Done – pridanie chýbajúceho prechodu medzi stavmi tasku
- 9571 – Analýza aktuálneho stavu – analýza momentálneho stavu ZMP v projekte
- 9573 – Spracovanie zápisnice č. 6.0 – vypracovanie zápisnice 6.0
- 9574 – Spracovanie zápisnice č. 6.1 – vypracovanie zápisnice 6.1

- 9575 – Spracovanie zápisnice č. 6.2 – vypracovanie zápisnice 6.2
- 9576 – Spracovanie zápisnice č. 7.0 – vypracovanie zápisnice 7.0
- 9577 – Spracovanie zápisnice č. 7.1 – vypracovanie zápisnice 7.1
- 9578 – Spracovanie zápisnice č. 7.2 – vypracovanie zápisnice 7.2
- 9579 – Analýza zisťovania polohy a orientácie hráča – analýza spôsobu, akým sa momentálne hráč orientuje na ihrisku
- 9580 – Metodika písania a commitovania kódu – spracovanie metodiky písania a commitovania kódu
- 9581 – Metodika testovania tasku – spracovanie metodiky testovania taskov
- 9582 – Metodika komunikácie – spracovanie metodiky komunikácie v tíme
- 9583 – Metodika review kódu – spracovanie metodiky review kódu
- 9584 – Metodika pracovania s TFS – spracovanie metodiky pracovania s TFS
- 9585 – Export taskov pre šprint č. 1 – export taskov z TFS pre použitie v sumarizácii šprintu
- 9587 – Spracovanie špecifikácie user story šprintu č. 2 – špecifikovanie user stories pridaných do šprintu č. 2
- 9588 – Spracovanie metodík – úprava a spojenie metodík do jedného dokumentu
- 9639 – Spracovanie retrospektívy z 1. šprintu – vypracovanie retrospektívy pre použitie v sumarizácii šprintu
- 9640 – Pridanie atribútov na task – pridanie atribútov, ktoré určujú, kto bude testovať a revidovať task
- 9641 – Nastavenie notifikovania Slacku podľa tagu – pridanie pravidiel pre posielanie notifikácií o zmene stavu tasku do kanálov na základe otagovania tasku menom
- 9643 – Analýza kalmanovho filtra hráča – analýza aktuálneho stavu a možného vylepšenia Kalmanovho filtra v projekte
- 9645 – Vytvorenie Backlog Taskov na základe pripomienok analýzy v yourkit – zadanie nových úloh do backlogu, ktoré vyplývajú z analýzu pomocou Yourkitu

2.2.4. Zhodnotenie

V tomto dlhšom šprinte sa nám podarilo dokončiť všetku zadanú prácu, na základe ktorej sme mohli aj vytvoriť user stories do ďalšieho šprintu. Išlo hlavne o optimalizáciu problémových častí kódu, vytvorenie metodík pre budúcu tímovú prácu a ďalšie analýzy funkčnosti projektu.

2.2.4.1. Podiel práce členov tímu

Podľa vykazovania práce na taskoch bola vytvorená tabuľka podielov práce jednotlivých členov tímu.

Člen tímu	Formálne stretnutia	Neformálne stretnutia	Práca na taskoch
Bc. Filip Dian	6h	8h	8,5h
Bc. Christopher Liebe	6h	8h	3,5h
Bc. Andrej Kútny	6h	8h	4,5h
Bc. Ľubomir Fischer	6h	8h	6h
Bc. Adriana Beňovičová	6h	8h	9h
Bc. Kitti Nagyová	6h	8h	6h
Bc. Adam Mikolášek	6h	8h	7h
Bc. Zoltán Csengődy	6h	8h	4h

Tabuľka 7 Podiel práce v 2. šprinte

2.3. Šprint č. 3

2.3.1. Základné informácie

Začiatok: 14.11.2018

Koniec: 27.11.2018

2.3.2. Retrospektíva

Dňa 27.11. 2018 sme ukončili šprint č. 3. Po jeho uzavretí sme urobili retrospektívu, počas ktorej sa každý člen vyjadril, čo sa mu páčilo a nepáčilo a určili si body, ktoré sa budeme v budúcnosti snažiť splniť. Pri bodoch sme aj určili, kto sa postará o jeho splnenie a v niektorých prípadoch aj vytvorili task.

2.3.2.1. Čo sa podarilo:

- máme väčší prehľad v kóde projektu (Filip)
- dobrá práca na súhrnnej dokumentácii (Filip, Andrej, Adriana, Adam, Kitti, Zoltán)
- rozbeh implementačných úloh (Ľubomír)
- prehľadnejšia a lepšie fungujúca wiki (Ľubomír, Andrej)
- nájdenie užitočných vecí v dokončených taskoch (Andrej)
- väčšia snaha dotiahnuť prácu v šprinte (Chris)
- dobrá komunikácia počas šprintu (Adam, Kitti, Zoltán)
- začatie úloh v skorom štádiu šprintu (Kitti)

2.3.2.2. Čo by sa mohlo zlepšiť:

- práca na taskoch na poslednú chvíľu (Filip, Andrej, Adam, Kitti, Zoltán)
- problém v komunikácii o zadanej úlohe (Filip, Zoltán, Andrej)
- málo implementácie (Ľubomír)
- takmer žiadna komunikácia s predošlými tímami (Ľubomír)
- málo pripravených úloh (Adriana)
- nedokončenie naplánovanej práce (Kitti)
- pomalý review (Kitti, Zoltán)

2.3.2.3. Akčné body:

- priebežné dokončovanie taskov, skoršie začatie prác (všetci)
- viac komunikácie o spoločných taskoch (všetci)

- viac implementačných prác (všetci)
- viac komunikácie s predošlým tímom (všetci)
- pripraviť backlog aj v predstihu, pripraviť tasky (solution owner)
- lepší odhad náročnosti taskov (všetci)
- skorší review (všetci)
- publikovanie zápisníc na stránku max do konca dňa stretnutia (web master)

2.3.3. Práca tímu

Na začiatku šprintu sme identifikovali prácu, ktorá by mala byť dokončená v rámci šprintu č. 1. Na základe vytvorených user stories sme zadali jednotlivé úlohy pre ich splnenie. Väčšina úloh bola vytvorená na začiatku šprintu a niektoré sa doplnili neskôr.

2.3.3.1. User stories

Do šprintu boli zaradené nasledujúce user stories.

ID	Title	Effort	State
9561	Analýza tímu Austin Villa	5	Done
9965	Publikovanie dokumentov na web stránke	2	Done
10003	Súhrnná dokumentácia	3	Done
10010	Aktualizovanie Wiki	3	Done
10015	Optimalizácia podľa Yourkitu	5	Done
10021	Špecifikácie	3	Done
10022	Zápisnice	1	Done

Tabuľka 8 User stories v šprinte č. 3

2.3.3.1.1. Opis user stories:

- 9561 – Analýza tímu Austin Villa – analýza spôsobu, akým sa hráč tímu Austin Villa orientuje na ihrisku
- 9965 – Publikovanie dokumentov na web stránke – nahratie chýbajúcich dokumentov na web stránku
- 10003 – Súhrnná dokumentácia – spracovanie súhrnnej dokumentácie riadenia potrebnej pre kontrolný bod po 9 týždňoch
- 10010 – Aktualizovanie Wiki – prekopanie dizajnu wiki

- 10015 – Optimalizácia poďla Yourkitu – skontrolovať kód, ktorý je podľa Yourkitu časovo náročný
- 10021 – Špecifikácie – spracovanie špecifikácií user stories
- 10022 – Zápisnice – spracovanie zápisníc zo stretnutí

2.3.3.2. Úlohy

Počas šprintu boli identifikované nasledujúce úlohy.

ID	Title	Assigned To	Priority	State	Completed Work
9562	Analýza videnia hráča Austin Villa	Bc. Andrej Kútny	2	Done	3
9763	Spracovanie sumarizácie šprintu č. 1	Bc. Filip Dian	2	Done	2
9827	Definícia rizík	Bc. Adriana Benovičová	2	Done	4
9831	Pozrieť sa na HighSkills	Bc. Filip Dian	2	Done	0,5
9832	Pozrieť sa na geometrické výpočty	Bc. Adam Mikolášek	2	Done	3
9915	Pozrieť sa na AgentModel	Bc. Filip Dian	2	Done	1,5
10004	Metodika mergovania kódu	Bc. Kitti Nagyová	2	Done	2
10012	Prekopať dizajn	Bc. Christopher Liebe	2	Done	1
10019	Súhrnný dokument po 9. týždni	Bc. Kitti Nagyová	2	Done	3
10020	Všeobecná šablóna pre dokumenty	Bc. Andrej Kútny	2	Done	1
10023	Napísať špecifikácie user stories	Bc. Ľubomír Fischer	2	Done	3
10024	Zápisnica č. 8.1	Bc. Andrej Kútny	2	Done	0,5
10025	Zápisnica č. 8.2	Bc. Andrej Kútny	2	Done	0,5
10026	Zápisnica č. 9.0	Bc. Filip Dian	2	Done	0,5
10027	Zápisnica č. 9.1	Bc. Andrej Kútny	2	Done	0,5

10028	Zápisnica č. 9.2	Bc. Andrej Kútny	2	Done	0,5
10029	Zápisnica č. 8.0	Bc. Adam Mikolášek	2	Done	0,5
10033	Automatická detekcia nových zápisníc	Bc. Christopher Liebe	2	Done	2
10117	Sumarizácia 2. šprintu	Bc. Filip Dian	1	Done	2

Tabuľka 9 Úlohy v šprinte č. 2

2.3.3.2.1. Opis úloh:

- 9562 – Analýza videnia hráča Austin Villa – analýza spôsobu získavania informácií, na základe ktorých sa hráč orientuje na ihrisku
- 9763 – Spracovanie sumarizácie šprintu č. 1 – publikovanie spracovanej sumarizácie 1. šprintu
- 9827 – Definícia rizík – opis rizík, ktoré vznikajú počas práce na projekte
- 9831 – Pozrieť sa na HighSkills – kontrola náročnosti metód v triede a optimalizácia v prípade potreby
- 9832 – Pozrieť sa na geometrické výpočty – kontrola náročnosti geometrických výpočtov a optimalizácia v prípade potreby
- 9915 – Pozrieť sa na AgentModel - kontrola náročnosti metód v triede a optimalizácia v prípade potreby
- 10004 – Metodika mergovania kódu – spracovanie metodiky popisujúcej mergovanie kódu
- 10012 – Prekopať dizajn – nasadenie nového dizajnu wiki (doteraz tam žiadny nebol)
- 10019 – Súhrnný dokument po 9. týždni – spracovanie súhrnnej dokumentácie o riadení projektu potrebnej ku kontrolnému bodu
- 10020 – Všeobecná šablóna pre dokumenty – spracovanie jednotného vzoru dokumentov
- 10023 – Napísať špecifikácie user stories – spracovať špecifikácie user stories, na základe ktorých sa vytvoria tasky
- 10024 – Zápisnica č. 8.1 – spracovanie zápisnice
- 10025 – Zápisnica č. 8.2 – spracovanie zápisnice
- 10026 – Zápisnica č. 9.0 – spracovanie zápisnice
- 10027 – Zápisnica č. 9.1 – spracovanie zápisnice

- 10028 – Zápisnica č. 9.2 – spracovanie zápisnice
- 10029 – Zápisnica č. 8.0 – spracovanie zápisnice
- 10033 – Automatická detekcia nových zápisníc – zjednodušenie nahrávania zápisníc na web stránku
- 10017 – Sumarizácia 2. šprintu – spracovanie informácií o druhom šprinte

2.3.4. Zhodnotenie

V tomto šprinte sa nám z dôvodu prác na iných predmetoch podarilo dokončiť len približne 2/3 práce. Väčšinou išlo o nedokončené alebo nerevidované/neotestované implementačné úlohy, ktoré sa tým pádom presunuli do ďalšieho šprintu.

2.3.4.1. Podiel práce členov tímu

Podľa vykazovania práce na taskoch bola vytvorená tabuľka podielov práce jednotlivých členov tímu. V celkovom počte hodín práce na taskoch je zarátaná aj práca strávená revidovaním a testovaním. Tasky, ktoré sa presunuli do ďalšieho šprintu zarátané nie sú.

Člen tímu	Formálne stretnutia	Neformálne stretnutia	Práca na taskoch
Bc. Filip Dian	6h	8h	7h
Bc. Christopher Liebe	6h	8h	3h
Bc. Andrej Kútny	6h	8h	6,5h
Bc. Ľubomir Fischer	6h	8h	4h
Bc. Adriana Beňovičová	6h	8h	5h
Bc. Kitti Nagyová	6h	8h	9h
Bc. Adam Mikolášek	6h	8h	3,5h
Bc. Zoltán Csengődy	6h	8h	2h

Tabuľka 10 Podiel práce v 3. šprinte

2.4. Šprint č. 4

2.4.1. Základné informácie

Začiatok: 28.11.2018

Koniec: 12.12.2018

2.4.2. Retrospektíva

Dňa 12.11.2018 sme ukončili šprint č. 4. Po jeho uzavretí sme urobili retrospektívu, počas ktorej sa každý člen vyjadril, čo sa mu páčilo a nepáčilo a určili si body, ktoré sa budeme v budúcnosti snažiť splniť. Pri bodoch sme aj určili, kto sa postará o jeho splnenie a v niektorých prípadoch aj vytvorili task.

2.4.2.1. Čo sa podarilo

- pokračovanie v implementáciách (Filip, Kitti, Ľubomír)
- práca začína mať reálne výsledky (Kitti)
- pomoc pri úlohách v tíme (Kitti, Zoltán, Adam)
- stretnutie aj mimo oficiálnych stretnutí (Andrej, Adam)
- dobrá komunikácia (Adam)
- stihnutie väčšiny práce (Zoltán)

2.4.2.2. Čo by sa mohlo zlepšiť

- dokončovanie taskov na konci šprintu (Kitti, Ľubomír, Zoltán)
- nedokončenie niektorých úloh (Filip, Zoltán)
- nespracovanie opisu zmien implementačných taskov (Filip, Zoltán)
- málo kontroly stavu úloh v TFS (Ľubomír)
- málo prác na projekte z dôvodu zaťaženia inými predmetmi (Andrej)
- nevyužívanie metodík (Adam)

2.4.2.3. Akčné body

- snažiť sa dokončovať úlohy priebežne, najlepšie v prvej polovici šprintu (všetci)
- písanie opisov po prvotnom dokončení tasku (všetci)
- častejšie kontrolovať stav práce v TFS (všetci)
- lepšie dodržovanie metodík, prípadne ich aktualizácia (všetci)

2.4.3. Práca tímu

Na začiatku šprintu sme identifikovali prácu, ktorá by mala byť dokončená v rámci šprintu č. 3. Na základe vytvorených user stories sme zadali jednotlivé úlohy pre ich splnenie. Väčšina úloh bola vytvorená na začiatku šprintu a niektoré sa doplnili neskôr.

2.4.3.1. User stories

Do šprintu boli zaradené nasledujúce user stories.

ID	Title	Effort	State
8959	Optimalizácia Kalmana	3	Done
8960	Implementácia BP Barabasa	8	Done
9830	Zámena súradníc X a Y	3	Done
10005	Synchronizácia vlákien	8	Done
10013	Vylepšenie zero moment pointu	8	Done
10269	Opisy zmien po optimalizácii	2	Done
10270	Optimalizácia podľa Yourkitu	3	Done
10271	Úprava zobrazovania stránky	2	Done
10274	Kontrola a doplnenie Wiki	2	Done
10277	Špecifikácie	2	Done
10318	Dokončenie funkcií pre orientáciu hráča na ihrisku	5	Approved
10319	Spracovanie súhrnnej dokumentácie a prezentácie	3	Approved
10457	Zápisnice	1	Done

Tabuľka 11 User stories v šprinte č. 4

2.4.3.1.1. Opis user stories

- 8959 – Optimalizácia Kalmana – prekontrolovanie správnosti rovníc Kalmanovho filtra v projekte a návrhy na optimalizáciu
- 8960 – Implementácia BP Barabasa – analýza a doimplementovanie funkcií z BP iného študenta, ktorý pracoval na vylepšení hráča
- 9830 – Zámena súradníc X a Y – identifikovanie miest v kóde, kde sa počíta s prehodenými súradnicami
- 10005 – Synchronizácia vlákien – pridanie synchronizácie medzi vláknami, ktoré boli pridané kvôli optimalizácii spracovania komunikácie

- 10013 – Vylepšenie zero moment pointu – zistenie, ako zlepšiť aktuálne implementovaný ZMP
- 10269 – Opisy zmien po optimalizácii – vypracovanie chýbajúcich dokumentov ku zmenám, ktoré boli spravené pre optimalizáciu spracovania komunikácie
- 10270 – Optimalizácia podľa Yourkitu – optimalizovanie metód, ktoré sú podľa Yourkitu výpočtovo náročné
- 10271 – Úprava zobrazovania stránky – vylepšenie dizajnu stránky
- 10274 – Kontrola a doplnenie Wiki – identifikovanie chýbajúcich analýz a ich dodatočné nahratie
- 10277 – Špecifikácie – vypracovanie špecifikácií user stories
- 10318 – Dokončenie funkcií pre orientáciu hráča na ihrisku – implementovanie funkcií, ktoré boli predošlým tímom navrhnuté, ale neimplementované
- 10319 – Spracovanie súhrnnej dokumentácie a prezentácie – vypracovanie dokumentov ku kontrolnému bodu po skončení semestra
- 10457 – Zápisnice

2.4.3.2. Úlohy

Počas šprintu boli identifikované nasledujúce úlohy.

ID	Title	Assigned To	Priority	State	Completed Work
9829	Pozrieť sa na Parser	Bc. Filip Dian	1	Done	0,5
9914	Pozrieť sa na LowSkills	Bc. Zoltán Csengődy	1	Done	6
10002	Kontrola osí x/y	Bc. Ľubomír Fischer	2	Done	4
10006	Kontrola rovníc z analýzy v kóde	Bc. Adam Mikolášek	2	Done	6
10007	Synchronizácia komunikačného vlákna	Bc. Christopher Liebe	1	Done	2
10008	Opis zmien po optimalizácií logov a správ do TF	Bc. Filip Dian	2	Done	1

10011	Úprava zobrazovania dokumentov na stránke	Bc. Zoltán Csengődy	2	Done	6
10014	Analýza, čo sa používa v kóde	Bc. Andrej Kútny	2	Done	2
10272	Validácia a optimalizácia ZMP	Bc. Andrej Kútny	2	Done	2
10273	Analýza súčasného stavu	Bc. Adriana Beňovičová	2	Done	4
10275	Kontrola, čo chýba	Bc. Kitti Nagyová	2	Done	1,5
10276	Doplniť, čo chýba	Bc. Kitti Nagyová	2	Done	2
10314	Vypracovanie špecifikacií	Bc. Ľubomír Fischer	1	Done	2
10321	Funkcia isIntersection	Bc. Filip Dian	1	Done	6
10322	Funkcia isT	Bc. Kitti Nagyová	1	Done	3
10323	Funkcia isPlus	Bc. Zoltán Csengődy	1	Done	6
10324	Funkcia isGoalBox	Bc. Adam Mikolášek	1	Done	2
10325	Výstup po 11.týždni - dokumentácia riadenia	Bc. Kitti Nagyová	1	In Progress	
10328	Výstup po 11.týždni - inžinierske dielo	Andrej Kútny	1	In Progress	
10382	vybavenie licencie pre bitbucket pre kod nasej webstranky	Bc. Kitti Nagyová	2	Done	1
10396	Upload stránky na git	Bc. Christopher Liebe	2	Done	1

10397	Metodika nahrávania obrázkov na wiki	Bc. Christopher Liebe	2	Done	0,5
10454	Sumarizácia šprintu č. 3	Bc. Filip Dian	1	Done	2
10458	Zápisnica 12.0	Bc. Zoltán Csengődy	2	Done	1
10459	Zápisnica 11.0	Bc. Christopher Liebe	2	Done	0,5
10532	Sumarizácia šprintu č. 4	Bc. Filip Dian	1	In Progress	

Tabuľka 12 Úlohy v šprinte č. 2

2.4.3.2.1. Opis úloh

- 9829 – Pozrieť sa na Parser – skontrolovať triedu, ktorej metódy sú podľa Yourkitu výpočtovo náročné a prípadne ich optimalizovať
- 9914 – Pozrieť sa na LowSkills - – skontrolovať triedu, ktorej metódy sú podľa Yourkitu výpočtovo náročné a prípadne ich optimalizovať
- 10002 – Kontrola osí x/y – nájdenie častí kódu, kde sú výpočty s prehodenými súradnicami
- 10006 – Kontrola rovníc z analýzy v kóde – skontrolovanie správnosti implementovaných rovníc Kalmanovho filtra v projekte
- 10007 – Synchronizácia komunikačného vlákna – implementácia synchronizácie vlákien, ktoré boli pridané pri optimalizácií spracovania komunikácie
- 10008 – Opis zmien po optimalizácií logov a správ do TF – spracovanie opisu zmien vykonaných pre optimalizáciu spracovania komunikácie
- 10011 – Úprava zobrazovania dokumentov na stránke – vylepšenie dizajnu stránky
- 10014 – Analýza, čo sa používa v kóde – skontrolovanie aktuálneho stavu implementácie ZMP
- 10272 – Validácia a optimalizácia ZMP – návrh na implementáciu vylepšení ZMP
- 10273 – Analýza súčasného stavu – kontrola stavu implementácie BP Barabása v projekte

- 10275 – Kontrola, čo chýba – identifikovanie chýbajúcich dokumentov, ktoré by mali byť na wiki
- 10276 – Doplniť, čo chýba – doplnenie chýbajúcich dokumentov na wiki
- 10314 – Vypracovanie špecifikácií – spracovanie špecifikácií user stories
- 10321 – Funkcia isIntersection – implementovanie funkcie, ktorá zisťuje, či sa dve úsečky pretínajú
- 10322 – Funkcia isT – implementovanie funkcie, ktorá zisťuje, či dve úsečky tvoria vzor T
- 10323 – Funkcia isPlus – implementovanie funkcie, ktorá zisťuje, či dve úsečky tvoria vzor +
- 10324 – Funkcia isGoalBox – kontrola a doimplementovanie funkcie, ktorá zisťuje, či dané čiary tvoria bránkovisko
- 10325 – Výstup po 11. týždni – dokumentácia riadenia – vypracovanie dokumentácie riadenia po kontrolnom bode na konci semestra
- 10328 – Výstup po 11. týždni – inžinierske dielo – vypracovanie inžinierskeho diela
- 10382 – Vybavenie licencie pre Bitbucket pre kód našej webstránky – vybavenie licencie na Bitbucket, aby sa dal verziovať kód webovej stránky tímu
- 10396 – Upload stránky na git – nahratie kódu do verziovacieho systému
- 10397 – Metodika nahrávania obrázkov na wiki – spracovanie metodiky akým spôsobom treba nahrávať obrázky na wiki
- 10454 – Sumarizácia šprintu č. 3 – spracovanie sumarizácie po 3. šprinte
- 10458 – Zápisnica 12.0 – spracovanie zápisnice z 12. stretnutia
- 10459 – Zápisnica 11.0 – spracovanie zápisnice z 11. stretnutia
- 10532 – Sumarizácia šprintu č. 4 – spracovanie sumarizácie po 4. šprinte

2.4.4. Zhodnotenie

V tomto šprinte sa nám podarilo dokončiť úlohy, ktoré sa presunuli z predošlého šprintu a podarilo sa nám splniť väčšinu implementačných úloh, ktoré boli prioritou. Niektoré úlohy sa Pár úloh, z dôvodu nedokončeného review alebo testovania sa nám presunuli do ďalšieho šprintu. Na jeho konci bolo prioritou vypracovať súhrnné dokumenty po kontrolnom bode na konci semestra.

2.4.4.1. Podiel práce členov tímu

Podľa vykazovania práce na taskoch bola vytvorená tabuľka podielov práce jednotlivých členov tímu. V celkovom počte hodín práce na taskoch je zarátaná aj práca strávená revidovaním a testovaním. Tasky, ktoré sa presunuli do ďalšieho šprintu zarátané nie sú.

Člen tímu	**Formálne stretnutia**	**Neformálne stretnutia**	**Práca na taskoch**
Bc. Filip Dian	6h	8h	11h
Bc. Christopher Liebe	6h	8h	5h
Bc. Andrej Kútny	6h	8h	6,5h
Bc. Ľubomir Fischer	6h	8h	7h
Bc. Adriana Beňovičová	6h	8h	4,5h
Bc. Kitti Nagyová	6h	8h	13h
Bc. Adam Mikolášek	6h	8h	12,5h
Bc. Zoltán Csengődy	6h	8h	20,5h

Tabuľka 13 Podiel práce v 4. šprinte

2.5. Šprint č. 5

2.5.1. Základné informácie

Začiatok: 13.12.2018

Koniec: 25.2.2019

2.5.2. Retrospektíva

Dňa 25.2.2019 sme ukončili šprint č. 5. Po jeho uzavretí sme urobili retrospektívu, počas ktorej sa každý člen vyjadril, čo sa mu páčilo a nepáčilo a určili si body, ktoré sa budeme v budúcnosti snažiť splniť. Pri bodoch sme aj určili, kto sa postará o jeho splnenie a v niektorých prípadoch aj vytvorili task.

2.5.2.1. Čo sa podarilo:

- nadviazanie na identifikovanú prácu v predošlom semestri (Filip)
- začalo sa oveľa viac implementovať (Filip)
- dobré plánovanie spoločnej práce – súvisiacich úloh (Andrej, Adam, Zoltán, Kitti, Ľubomír)
- dobrý prehľad v backlogu a dostatok určenej práce (Andrej)
- rýchly návrat do reality (Andrej, Adam)
- dobré rozdelenie úloh (Zoltán)
- identifikovanie relevantnej práce (Ľubomír, Adriana)
- dobrý rozbeh do semestra (Kitti, Adriana)

2.5.2.2. Čo by sa mohlo zlepšiť:

- nepresná user story a nedostatky v analýze (Filip, Adriana)
- neskoršie dokončovanie úloh (Filip, Kitti)
- pomalší začiatok šprintu (Andrej)
- veľa nedokončených úloh (Adam, Kitti)
- slabé zadefinovanie spoločných úloh (Zoltán)
- nedokončenie priradenej úlohy, vysoká zložitosť pre jedného člena (Ľubomír)
- neaktuálne informácie na wiki (Ľubomír)

2.5.2.3. Akčné body:

- lepšie zadefinovanie user story a viac konzultovania (Solution owner + všetci)

- rozdeľovanie zložitejších úloh, aby sa dali dokončiť v šprinte (Scrum master + Solution owner)
- lepší odhad množstva a zložitosti úloh na stretnutí (všetci)
- presnejší opis, kto má čo robiť v rámci úlohy (Solution owner)
- aktualizácia a oprava wiki – počítanie pozície (Web master)
- priebežné zapisovanie opisov user story (Solution owner)

2.5.3. Práca tímu

Na začiatku šprintu sme identifikovali prácu, ktorá by mala byť dokončená v rámci šprintu č. 5. Na základe vytvorených user stories sme zadali jednotlivé úlohy pre ich splnenie. Väčšina úloh bola vytvorená na začiatku šprintu a niektoré sa doplnili neskôr.

2.5.3.1. User stories

Do šprintu boli zaradené nasledujúce user stories.

ID	Title	Effort	State
10727	Zápisnice	1	Done
10495	Návrhy	3	Done
10536	Implementácia funkcií zisťovania vzorov čiar	8	Done (Split)
10537	Chýbajúce dokumentácie k úlohám	2	Done

Tabuľka 14 User stories v šprinte č. 5

2.5.3.1.1. Opis user stories:

- 10727 – Zápisnice – vypracovanie zápisníc
- 10495 – Návrhy – premyslenie možných spôsobov vylepšenia určovania polohy hráča
- 10536 - Implementácia funkcií zisťovania vzorov čiar – implementovanie identifikovaných metód na zisťovanie polohy hráča na ihrisku
- 10537 - Chýbajúce dokumentácie k úlohám – doplnenie chýbajúcich opisov dokončených úloh

2.5.3.2. Úlohy

Počas šprintu boli identifikované nasledujúce úlohy.

ID	Title	Assigned To	Priority	State	Completed Work

10009	Opis zmien po pridaní vlákien	Bc. Christopher Liebe	2	Done	0.5
10496	Volania justLinesSeen - pouzitie na vhodnych miestach	Bc. Filip Dian	2	Done	0.5
10497	Volania isPlus - pouzitie na vhodnych miestach	Bc. Zoltán Csengődy	2	Done	4
10498	Volania isGoalBox - pouzitie na vhodnych miestach	Bc. Adam Mikolášek	2	Done	3
10499	Volania isIntersection - pouzitie na vhodnych miestach	Bc. Filip Dian	2	Done	0.5
10500	Volania isT - pouzitie na vhodnych miestach	Bc. Kitti Nagyová	2	Done	2
10509	Doplnenie wiki sprintu 3,4 a 5	Bc. Kitti Nagyová	2	Done	2
10728	Doplniť chýbajúce zápisnice	Bc. Christopher Liebe	2	Done	1
10729	Zapisnica 13.0	Bc. Andrej Kútny	2	Done	0.5
10730	Zapisncia 13.1	Bc. Andrej Kútny	2	Done	0.5
10791	zapisnica 14.2	Bc. Kitti Nagyová	2	Done	1
10862	Zápisnica 14.0	Bc. Zoltan Csengődy	2	Done	0.5

Tabuľka 15 Úlohy v šprinte č. 5

2.5.3.2.1. Opis úloh:

- 10009 – Opis zmien po pridaní vlákien – doplnenie opisu zmien po pridaní komunikačných vlákien
- 10496 – Volania justLinesSeen- použitie na vhodných miestach – zistenie, kde sa bude využívať implementovaná metóda

- 10497 – Volania isPlus – použitie na vhodných miestach - zistenie, kde sa bude využívať implementovaná metóda
- 10498 – Volania isGoalBox – použitie na vhodných miestach - zistenie, kde sa bude využívať implementovaná metóda
- 10499 – Volania isIntersection – použitie na vhodných miestach - zistenie, kde sa bude využívať implementovaná metóda
- 10500 - Volania isT – použitie na vhodných miestach - zistenie, kde sa bude využívať implementovaná metóda
- 10509 – Doplnenie wiki šprintu 3, 4 a 5 – pridanie chýbajúcich dokumentov o šprinte na wiki
- 10728 – Doplniť chýbajúce zápisnice – pridanie chýbajúcich zápisníc na webstránku
- 10729 – Zápisnica 13.0
- 10730 – Zápisnica 13.1
- 10731 – Zápisnica 14.2
- 10862 – Zápisnica 14.0

2.5.4. Zhodnotenie

Šprint bol pokračovaním prác zo zimného semestra, takže prácu sme mali priradenú už vtedy a niektoré úlohy pribudli až počas neho. Prvý týždeň šprintu bol skôr „rozbehový“, keďže bol začiatok semestra. Veľa práce sa prenieslo do ďalšieho šprintu.

2.5.4.1. Podiel práce členov tímu

Podľa vykazovania práce na taskoch bola vytvorená tabuľka podielov práce jednotlivých členov tímu. V celkovom počte hodín práce na taskoch je zarátaná aj práca strávená revidovaním a testovaním. Tasky, ktoré sa presunuli do ďalšieho šprintu zarátané nie sú.

Člen tímu	Formálne stretnutia	Neformálne stretnutia	Práca na taskoch
Bc. Filip Dian	6h	8h	1h
Bc. Christopher Liebe	6h	8h	1,5h
Bc. Andrej Kútny	6h	8h	1h
Bc. Ľubomir Fischer	6h	8h	0h
Bc. Adriana Beňovičová	6h	8h	0h

Bc. Kitti Nagyová	6h	8h	5h
Bc. Adam Mikolášek	6h	8h	3h
Bc. Zoltán Csengődy	6h	8h	4,5h

Tabuľka 16 Podiel práce v 5. šprinte

2.6. Šprint č. 6

2.6.1. Základné informácie

Začiatok: 26.2.2019

Koniec: 11.3.2019

2.6.2. Retrospektíva

Dňa 11.3.2019 sme ukončili šprint č. 6. Po jeho uzavretí sme urobili retrospektívu, počas ktorej sa každý člen vyjadril, čo sa mu páčilo a nepáčilo a určili si body, ktoré sa budeme v budúcnosti snažiť splniť. Pri bodoch sme aj určili, kto sa postará o jeho splnenie a v niektorých prípadoch aj vytvorili task.

2.6.2.1. Čo sa podarilo

- nastavenie Mavenu (Filip)
- spracovanie analýzy testovania zisťovania vzorov čiar (Filip)
- spoločná práca na problémoch (Andrej, Kitti)
- spolupráca na úlohe otočenia osí (Ľubomír)
- dobrá vzájomná komunikácia (Adriana, Kitti)
- jednoduchšie nastavenie projektu (Chris)
- dokončenie dôležitých úloh (Zoltán)
- zlepšujúca sa tímová práca (Adam)

2.6.2.2. Čo by sa mohlo zlepšiť

- nedokončené viaceré úlohy (Filip, Ľubomír, Andrej)
- zle odhadnutá zložitosť niektorých úloh (Filip, Adam)
- nedodržiavanie postupnosti v zmenách stavov úloh (Ľubomír)
- neskorá práca na úlohách (Ľubomír, Kitti)
- očakávania prekonali plán (Chris, Andrej)
- málo priradených úloh (Adam)

2.6.2.3. Akčné body

- kontrolovanie zmien stavov úloh (scrum master + všetci)
- lepší odhad zložitosti úloh (všetci)
- skoršie rozpracovanie úloh (všetci)

- nezabúdať na scrum poker (scrum master + všetci)

2.6.3. Práca tímu

Na začiatku šprintu sme identifikovali prácu, ktorá by mala byť dokončená v rámci šprintu č. 6. Na základe vytvorených user stories sme zadali jednotlivé úlohy pre ich splnenie. Väčšina úloh bola vytvorená na začiatku šprintu a niektoré sa doplnili neskôr.

2.6.3.1. User stories

Do šprintu boli zaradené nasledujúce user stories.

ID	Title	Effort	State
10278	Prejdenie TODO	8	Done (split)
10508	Oprava osí x/y	8	Done
10853	Testovanie a vylepšenie identifikovania vzorov čiar	5	Done (split)
10858	Nastavenie Maven repozitárov do projektu	3	Done
10860	Zápisnice	1	Done

Tabuľka 17 User stories v šprinte č. 6

2.6.3.1.1. Opis user stories:

- 10278 – Prejdenie TODO – kontrola a vytvorenie úloh na základe identifikovaných TODO v kóde predošlými tímami
- 10508 – Oprava osí x/y – zámena osí do korektného stavu
- 10853 – Testovanie a vylepšenie identifikovania vzorov čiar – otestovanie implementovaných úloh na zisťovanie vzorov čiar a premyslenie vylepšení
- 10858 – Nastavenie Maven repozitárov do projektu – prekonvertovanie projektu do Maven formátu pre jednoduchšiu manipuláciu s dependencies
- 10860 – Zápisnice – vypracovanie zápisníc

2.6.3.2. Úlohy

Počas šprintu boli identifikované nasledujúce úlohy.

ID	Title	Assigned To	Priority	State	Completed Work
10315	Identifikácia relevantnej práce v TODO v kóde	Bc. Andrej Kútny	2	Done	4

10507	Otočenie osí x/y	Bc. Ľubomír Fischer	1	Done	6
10608	consider relativeDistanceToTarget, consider annotation's min_distance	Bc. Andrej Kútny	2	Done	2
10732	Analýza použitia funkcií zisťovania čiar	Bc. Adriana Beňovičová	1	Done	5
10840	Calculate if two lines are part of central circle	Bc. Zoltán Csengődy	2	Done	1
10847	Communication test not finished. Insert testing parameters	Bc. Kitti Nagyová	2	Done	2
10856	Kontrola metód s otáčaním	Bc. Filip Dian	1	Done	6
10859	Prekonvertovanie projektu do mavenu	Bc. Christopher Liebe	2	Done	2
10861	Zápisnica 15.0	Bc. Kitti Nagyová	2	Done	2
10978	Zapisnica 16.0	Bc. Zoltán Csengődy	2	Done	2
10979	Spracovanie šprintu č. 5	Bc. Filip Dian	2	Done	1
11024	Nahranie zápisníc č.15 a 16	Bc. Adriana Beňovičová	2	Done	0,5

Tabuľka 18 Úlohy v šprinte č. 6

2.6.3.2.1. Opis úloh

- 10315 – Identifikácia relevantnej práce v TODO v kóde – vytvorenie úloh na základe kontroly TODO v kóde
- 10507 – Otočenie osí x/y – implementácia spätného prehodenia osí do korektného stavu
- 10608 - consider relativeDistanceToTarget, consider annotation's min_distance – vypracovanie TODO

- 10732 – Analýza použitia funkcií zisťovania čiar – premyslieť, ako sa implementované funkcie použijú pri zisťovaní polohy hráča
- 10840 - Calculate if two lines are part of central circle – vypracovanie TODO
- 10847 - Communication test not finished. Insert testing parameters – vypracovanie TODO
- 10856 – Kontrola metód s otáčaním – nájdenie miest vo vyšších vrstvách, kde sú osi prehodené a oprava
- 10859 – Prekonvertovanie projektu do Mavenu – skonvertovanie projektu do Maven formátu
- 10861 - Zápisnica 15.0 – vypracovanie zápisnice
- 10978 – Zápisnica 16.0 – vypracovanie zápisnice
- 10979 – Spracovanie šprintu č. 5 – vypracovanie zhrnutia šprintu č. 6
- 11024 – Nahranie zápisníc č.15 a 16 – doplnenie chýbajúcich zápisníc na web stránku

2.6.4. Zhodnotenie

V šprinte sme sa sústredili na dokončenie implementácie zisťovania polohy hráča na základe videných vzorov čiar a jej testovania. Testovanie a dokončenie analýzy vylepšenia sa preniesli do ďalšieho šprintu, čo sa dalo očakávať pri pomerne veľkej komplexnosti user story. Okrem toho sme začali vypracovávať identifikované TODO predošlých tímov, kde sa takisto v rámci šprintu nestihlo všetko.

2.6.4.1. Podiel práce členov tímu

Podľa vykazovania práce na taskoch bola vytvorená tabuľka podielov práce jednotlivých členov tímu. V celkovom počte hodín práce na taskoch je zarátaná aj práca strávená revidovaním a testovaním. Tasky, ktoré sa presunuli do ďalšieho šprintu zarátané nie sú.

Člen tímu	Formálne stretnutia	Neformálne stretnutia	Práca na taskoch
Bc. Filip Dian	6h	8h	7h
Bc. Christopher Liebe	6h	8h	2h
Bc. Andrej Kútny	6h	8h	6h
Bc. Ľubomir Fischer	6h	8h	6h
Bc. Adriana Beňovičová	6h	8h	5,5h

Bc. Kitti Nagyová	6h	8h	4h
Bc. Adam Mikolášek	6h	8h	3h
Bc. Zoltán Csengődy	6h	8h	3h

Tabuľka 19 Podiel práce v 6. šprinte

2.7. Šprint č. 7

2.7.1. Základné informácie

Začiatok: 12.3.2019

Koniec: 25.3.2019

2.7.2. Retrospektíva

Dňa 25.3.2019 sme ukončili šprint č. 7. Po jeho uzavretí sme urobili retrospektívu, počas ktorej sa každý člen vyjadril, čo sa mu páčilo a nepáčilo a určili si body, ktoré sa budeme v budúcnosti snažiť splniť. Pri bodoch sme aj určili, kto sa postará o jeho splnenie a v niektorých prípadoch aj vytvorili task.

2.7.2.1. Čo sa podarilo

- práca na TODO, viac implementácie (Filip, Ľubomír, Andrej)
- úspešné testovanie vzorov čiar (Filip)
- proaktívne prideľovanie práce (Ľubomír)
- komunikácia o úlohách (Ľubomír, Kitti)
- práca Scrum Mastera (Adriana)
- upresnenie výstupu úlohy (Adriana)
- rýchla odozva zo Slacku (Zoltán)
- robustný backlog (Andrej)
- rýchly feedback od členov tímu (Andrej)
- pripomenutie review (Andrej)

2.7.2.2. Čo by sa mohlo zlepšiť

- nepublishovanie úloh v tomto stave načas (Filip)
- slabé sledovanie svojich review (Filip)
- rozbitie Testframeworku (Filip, Kitti, Andrej, Adam)
- lepšie otestovanie dopadov implementácií TODO (Ľubomír)
- neskoré review (Zoltán)
- zlá úroveň kódu a chýbajúca dokumentácia (Chris, Andrej, Adam)
- neskoré dokončovanie úloh (Kitti)
- slabý time management (Andrej)
- veľa drobnej práce v úlohách (Andrej) //akčný bod: robiť a menej sa sťažovať

- rozptylovanie nesprávne napísaným kódom mimo úlohy

2.7.2.3. Akčné body

- dôkladnejšie otestovanie vlastných implementácií (všetci)
- treba sa ozvať v prípade, že niečo nefunguje (všetci)
- dôkladnejšie sledovanie pridelených review (všetci)
- zredukovanie notifikácií zo Slacku (Filip)
- dôkladne doplniť získanú dokumentáciu na Wiki (všetci)
- vyčistiť kód podľa TODO (všetci)
- spísanie stavu a možnej budúcej práce pre hladší rozbeh ďalšieho tímu (všetci)
- opraviť Testframework (Filip + Chris)
- sústredenie sa na náplň úlohy (všetci)

2.7.3. Práca tímu

Na začiatku šprintu sme identifikovali prácu, ktorá by mala byť dokončená v rámci šprintu č. 7. Na základe vytvorených user stories sme zadali jednotlivé úlohy pre ich splnenie. Väčšina úloh bola vytvorená na začiatku šprintu a niektoré sa doplnili neskôr.

2.7.3.1. User stories

Do šprintu boli zaradené nasledujúce user stories.

ID	Title	Effort	State
11054	Vypracovanie TODO	8	Done (split)
11055	Testovanie a vylepšenie identifikovania vzorov čiar	5	Done
11057	Zápisnice	1	Done
11173	Aktualizovanie wiki s novo implementovanými funkciami	2	Done

Tabuľka 20 User stories v šprinte č. 7

2.7.3.1.1. Opis user stories:

- 11054 – Vypracovanie TODO – implementácia a oprava identifikovaných TODO
- 11055 – Testovanie a vylepšenie identifikovania vzorov čiar – vykonanie testovania predošle implementovaných funkcií na zisťovanie vzorov čiar
- 11057 – Zápisnice – Spracovanie zápisníc

- 11173 – Aktualizovanie wiki s novo implementovanými funkciami – doplnenie chýbajúcej dokumentácie o implementovaných funkciách na zisťovanie vzorov čiar

2.7.3.2. Úlohy

Počas šprintu boli identifikované nasledujúce úlohy.

ID	Title	Assigned To	Priority	State	Completed Work
10327	Funkcia justLinesSeen	Bc. Filip Dian	1	Done	10
10844	effectors into HashMap	Bc. Christopher Liebe	2	Done	1
10846	Premyslieť, ako sa bude riešiť zlá nadväznosť pohybov	Bc. Ľubomír Fischer	2	Done	5
10849	Implement singleton in SkillsFromXmlLoader.java	Bc. Christopher Liebe	2	Done	1
10850	send an exit command to agents not launched by the test framework	Bc. Zoltán Csengődy	2	Done	1
10854	oprava funkcie isT	Bc. Adam Mikolášek	1	Done	3
10855	Opis testovania funkcie justLinesSeen	Bc. Adriana Beňovičová	1	Done	5
10983	Vytvorenie testovacej taktiky	Bc. Filip Dian	1	Done	8
10984	opraviť prípad, keď je size 0	Bc. Kitti Nagyová	2	Done	1
11058	Zápisnica 17.0	Bc. Lubomir Fischer	2	Done	1
11174	Doplniť informácie k funkciám vzorov	Bc. Zoltán Csengődy	2	Done	3

11175	Zápisnica 18.0	Bc. Zoltán Csengődy	2	Done	2
11210	Sumarizácia šprintu c. 6	Bc. Christopher Liebe	2	Done	1
11304	Zápisnica 19.0	Bc. Adriana Beňovičová	2	Done	1

Tabuľka 21 Úlohy v šprinte č. 7

2.7.3.2.1. Opis úloh

- 10327 – Funkcia justLinesSeen – implementácia funkcie na identifikovanie polohy hráča pomocou funkcií na zisťovanie vzorov čiar
- 10833 – effectors into HashMap – vypracovanie TODO
- 10846 – Premyslieť, ako sa bude riešiť zlá nadväznosť pohybov – vypracovanie TODO
- 10849 – Implement singleton in SkillFromXmlLoader.java – vypracovanie TODO
- 10850 – send an exit command to agents not launched by the test framework – vypracovanie TODO
- 10854 – oprava funkcie isT – úprava nesprávnej logiky
- 10855 – Opis testovania funkcie justLinesSeen – vypracovanie dokumentu ako sa bude testovať implementovaná funkcia
- 10984 – Vytvorenie testovacej taktiky – implementovanie taktiky, ktorá sa použije na účely testovania funkcie justLinesSeen
- 11058 – Zápisnica 17.0 – vypracovanie zápisnice
- 11173 – Doplniť informácie k funkciám vzorov – aktualizovanie wiki, kde boli funkcie opísané ako neimplementované
- 11175 – Zápisnica 18.0 – vypracovanie zápisnice
- 11210 – Sumarizácia šprintu č. 6 – vypracovanie sumarizácie šprintu
- 11304 – Zápisnica 18.0 – vypracovanie zápisnice

2.7.4. Zhodnotenie

V šprinte sme sa sústredili na otestovanie a doladenie funkcií na identifikovanie polohy hráča a správne zdokumentovanie. Taktiež sa podarilo vyriešiť niektoré TODO, z ktorých nám však stále ostalo veľa zmien, ktoré sa presunuli do ďalšieho šprintu. Aj spolu s vedúcim projektu

sme sa zhodli, že kód a dokumentácia nie sú v najlepšom stave a tak snaha ďalších šprintov bude poupratovať kód a rozšíriť stávajúcu dokumentáciu, aby ďalšie tímy mali jednoduchší nábeh na projekt. V rámci snahy bude aj vytvorený obsiahlejší dokument o aktuálnom stave hráča a návrhoch, ako ho zlepšiť.

2.7.4.1. Podiel práce členov tímu

Podľa vykazovania práce na taskoch bola vytvorená tabuľka podielov práce jednotlivých členov tímu. V celkovom počte hodín práce na taskoch je zarátaná aj práca strávená revidovaním a testovaním. Tasky, ktoré sa presunuli do ďalšieho šprintu zarátané nie sú.

Člen tímu	Formálne stretnutia	Neformálne stretnutia	Práca na taskoch
Bc. Filip Dian	6h	8h	18h (split)
Bc. Christopher Liebe	6h	8h	3h
Bc. Andrej Kútny	6h	8h	1h (split)
Bc. Ľubomir Fischer	6h	8h	7h
Bc. Adriana Beňovičová	6h	8h	6h
Bc. Kitti Nagyová	6h	8h	5h
Bc. Adam Mikolášek	6h	8h	5h
Bc. Zoltán Csengődy	6h	8h	8h

Tabuľka 22 Podiel práce v 7. šprinte

2.8. Šprint č. 8

2.8.1. Základné informácie

Začiatok: 26.3.2019

Koniec: 8.4.2019

2.8.2. Retrospektíva

Dňa 8.4.2019 sme ukončili šprint č. 8. Po jeho uzavretí sme urobili retrospektívu, počas ktorej sa každý člen vyjadril, čo sa mu páčilo a nepáčilo a určili si body, ktoré sa budeme v budúcnosti snažiť splniť. Pri bodoch sme aj určili, kto sa postará o jeho splnenie a v niektorých prípadoch aj vytvorili task.

2.8.2.1. Čo sa podarilo

- doplňovanie dokumentov na wiki (Filip)
- dobré rozdelenie úloh (Filip)
- identifikovanie cieľa do konca semestra na širšie zdokumentovanie a vyčistenie kódu (Andrej, Adam)
- dostatok úloh v backlogu šprintu (Andrej)
- silný team spirit (Andrej, Adriana, Zoltán, Adam)
- spracovanie dokumentácie pre ďalšie tímy (Kitti)
- komunikácia (Kitti)
- nálada pondelkových stretnutí (Kitti, Adriana)
- vyjasnenie výstupu projektu (Adriana)
- Chris z časti pokračuje v práci aj keď už nie je študent (Zoltán)

2.8.2.2. Čo by sa mohlo zlepšiť

- neskoré rozpracovanie úloh (Filip, Andrej, Kitti, Adam)
- ťažkopádne vypracovanie TODO z dôvodu málo informácií (Filip)
- neskoré review (Filip)
- Chris by mohol prestávať odchádzať z tímu (Andrej, Kitti)
- pozabudnutie na odovzdávanie kontrolnej dokumentácie (Andriana)
- nekonzistentné ukladanie dokumentov (Adriana)
- v TFS je veľa nepriradených úloh a prehadzujú sa do ďalšieho šprintu (Zoltán)

2.8.2.3. Akčné body

- ukladanie dokumentov aj do Doc záložky v TFS (všetci)
- upratanie úloh v TODO a prehodnotenie, čo sa vždy do šprintu pridá (Scrum Master)
- skoršie review a rozpracovanie úloh (všetci)

2.8.3. Práca tímu

Na začiatku šprintu sme identifikovali prácu, ktorá by mala byť dokončená v rámci šprintu č. 8. Na základe vytvorených user stories sme zadali jednotlivé úlohy pre ich splnenie. Väčšina úloh bola vytvorená na začiatku šprintu a niektoré sa doplnili neskôr.

2.8.3.1. User stories

Do šprintu boli zaradené nasledujúce user stories.

ID	Title	Effort	State
10395	Prehľadnosť kódu	8	Done (split)
11270	Oprava mena tímu	1	Done
11271	Oprava TestFrameworku	5	Done (split)
11277	Kontrola Wiki	2	Done
11279	Zápisnice	1	Done
11331	Pokračovanie TODO	8	Done (split)

Tabuľka 23 User stories v šprinte č. 8

2.8.3.1.1. Opis user stories:

- 10395 – Prehľadnosť kódu – kontrola a vyčistenie kódu od nepotrebných importov, metód a súborov
- 11270 – Oprava mena tímu – aktualizovanie starého mena tímu (predošlého)
- 11271 – Oprava TestFrameworku – oprava nefungujúceho pridávanie hráča na ihrisko
- 11277 – Kontrola Wiki – doplnenie chýbajúcich analýz
- 11279 – Zápisnice – vypracovanie zápisníc
- 11331 – Pokračovanie TODO – pokračovanie vypracovávania TODO

2.8.3.2. Úlohy

Počas šprintu boli identifikované nasledujúce úlohy.

ID	Title	Assigned To	Priority	State	Completed Work
10843	Compute relative fitness to strength of kick (annotations)	Bc. Adam Mikolášek	2	Done	2.5
10845	refactor into class hierarchy	Bc. Andrej Kútny	2	Done	2
10848	better check two lists ? Maybe HASH ?	Bc. Andrej Kútny	2	Done	8
10851	Spracovanie ostatných TODO	Bc. Adam Mikolášek	2	Done	4.5
11056	Vytvorenie logiky rozlíšenia vzorov	Bc. Adriana Beňovičová	2	Done	6.5
11272	Vymazať nepoužité importy	Bc. Zoltán Csengődy	1	Done	2
11273	Vymazat nepouzite metody 1	Bc. Filip Dian	1	Done	3
11274	Zjednotenie verzii knižníc v Mavene	Bc. Christopher Liebe	1	Done	2
11336	Change team name to WAWOT	Bc. Ľubomír Fischer	2	Done	1
11432	Zápisnica 20.0	Bc. Adam Mikolášek	2	Done	0.5
11440	Spracovanie šprintu c. 7	Bc. Filip Dian	2	Done	1
11460	Doplniť chýbajúce vytvorené analýzy časť 2	Bc. Adriana Beňovičová	2	Done	4
11517	Zápisnica 21.0	Bc. Kitti Nagyová	2	Done	0.5

Tabuľka 24 Úlohy v šprinte č. 8

2.8.3.2.1. Opis úloh

- 10843 – Compute relative fitness to strength of kick (annotations) – vypracovanie TODO
- 10845 – refactor into class hierarchy – vypracovanie TODO

- 10848 – better check two lists? maybe HASH? – vypracovanie TODO
- 10851 – Spracovanie ostatných TODO – nájdenie a opis ďalších TODO
- 11056 – Vytvorenie logiky rozlíšenia vzorov – premyslenie, ako sa bude hráč orientovať podľa identifikovania vzorov čiar
- 11272 – Vymazať nepoužité importy – vymazanie nepoužitých importov
- 11273 – Vymazať nepoužité metódy 1 – vymazanie nepoužitých metód v triedach s názvom od A po M
- 11274 – Zjednotenie verzií knižníc v Mavene – úprava pom.xml na jednotné nastavovanie verzií
- 11336 – Change team name to WAWOT - aktualizovanie starého mena tímu (predošlého)
- 11432 – Zápisnica 20.0 – vypracovanie zápisnice
- 11440 – Spracovanie šprintu č. 7 – spracovanie dokumentácie k predošlému šprintu
- 11460 – Doplniť chýbajúce vytvorené analýzy časť 2 – doplnenie Wiki o nové analýzy
- 11517 – Zápisnica 21.0 – vypracovanie zápisnice

2.8.4. Zhodnotenie

V šprinte sme pokračovali v pracovaní na TODO úlohách a začali sme postupne čistiť kód. Rovnako bola aj skontrolovaná Wiki a boli popridávané nedávno vytvorené analýzy, ktoré tam chýbali. Oprava Testframeworku sa zatiaľ nepodarila a bude sa v nej pokračovať ďalší šprint. Viac vecí sa nepodarilo dokončiť, keďže sme sa sústredili aj na vypracovanie kontrolnej dokumentácie k 9. týždňu.

2.8.4.1. Podiel práce členov tímu

Podľa vykazovania práce na taskoch bola vytvorená tabuľka podielov práce jednotlivých členov tímu. V celkovom počte hodín práce na taskoch je zarátaná aj práca strávená revidovaním a testovaním. Tasky, ktoré sa presunuli do ďalšieho šprintu zarátané nie sú.

Člen tímu	Formálne stretnutia	Neformálne stretnutia	Práca na taskoch
Bc. Filip Dian	6h	8h	4h
Bc. Christopher Liebe	6h	8h	2h
Bc. Andrej Kútny	6h	8h	10h
Bc. Ľubomir Fischer	6h	8h	1h

Bc. Adriana Beňovičová	6h	8h	10.5h
Bc. Kitti Nagyová	6h	8h	0.5h
Bc. Adam Mikolášek	6h	8h	7.5h
Bc. Zoltán Csengődy	6h	8h	2h

Tabuľka 25 Podiel práce v 8. šprinte

2.9. Šprint č. 9

2.9.1. **Základné informácie**

Začiatok: 9.4.2019

Koniec: 22.4.2019

Tím:

Bc. Adriana Beňovičová

Bc. Zoltán Csengődy

Bc. Filip Dian

Bc. Ľubomír Fischer

Bc. Andrej Kútny

Bc. Christopher Liebe

Bc. Adam Mikolášek

Bc. Kitti Nagyová

2.9.2. Retrospektíva

Dňa 22.4.2019 sme ukončili šprint č. 9. Po jeho uzavretí sme urobili retrospektívu, počas ktorej sa každý člen vyjadril, čo sa mu páčilo a nepáčilo a určili si body, ktoré sa budeme v budúcnosti snažiť splniť. Pri bodoch sme aj určili, kto sa postará o jeho splnenie a v niektorých prípadoch aj vytvorili task.

2.9.2.1. Čo sa podarilo

- doplňovanie dokumentov na Wiki (Filip)
- dobré rozdelenie úloh (Filip)
- identifikovanie cieľa do konca semestra na širšie zdokumentovanie a vyčistenie kódu (Andrej)
- dostatok vecí v backlogu (úlohy) šprintu (Andrej)
- silný team spirit (Andrej, Adriana, Zoltán, Adam)
- spracovanie dokumentácie pre ďalšie tímy (Kitti)
- dobrá komunikácia (Kitti)
- príjemné pondelkové stretnutia (Kitti, Adriana)
- ujasnenie výstupu projektu (Adriana)
- refactor kódu (Adam)

2.9.2.2. Čo by sa mohlo zlepšiť

- neskoré rozpracovanie úloh (Filip, Andrej, Kitti, Adam)
- zložité vypracovanie TODO z dôvodu málo informácií (Filip)
- neskoré review (Filip)
- Chris by mohol prestávať odchádzať z tímu (Andrej, Kitti)
- zabudnutie na odovzdanie kontrolnej dokumentácie (Adriana)
- rozhádzané ukladanie dokumentov (Adriana)
- v TFS je veľa nepriradených úloh, ktoré sa prehadzujú do ďalšieho šprintu (Zoltán)

2.9.2.3. Akčné body

- ukladanie dokumentov aj do Doc záložky v TFS, doplnenie chýbajúcich (všetci)
- upratanie úloh v TODO a prehodnotenie, čo sa vždy do šprintu pridá (Scrum Master + Solution owner)
- skoršie review a rozpracovanie úloh (všetci)

2.9.3. Práca tímu

Na začiatku šprintu sme identifikovali prácu, ktorá by mala byť dokončená v rámci šprintu č. 9. Na základe vytvorených user stories sme zadali jednotlivé úlohy pre ich splnenie. Väčšina úloh bola vytvorená na začiatku šprintu a niektoré sa doplnili neskôr.

2.9.3.1. User stories

Do šprintu boli zaradené nasledujúce user stories.

ID	Title	Effort	State
11269	Vyčistenie zbytočných súborov	3	Done
11280	Vypracovanie dokumentácie ku kontrolnému bodu	2	Done
11551	Dokončenie TODO	8	Done
11552	Zmazanie nepoužitých metód	3	Done
11553	Oprava pridávania hráča v Testframeworku	5	Done
11554	Doplnenie chýbajúcich analýz	1	Done
11576	Zápisnice	1	Done

Tabuľka 26 User stories v šprinte č. 9

2.9.3.1.1. Opis user stories

- 11269 – Vyčistenie zbytočných súborov – v projekte sa nachádza mnoho súborov, ktoré sa už nepoužívajú, sú vhodné na vymazanie

- 11280 – Vypracovanie dokumentácie ku kontrolnému bodu – spracovanie dokumentu k 9. týždňu
- 11551 – Dokončenie TODO – dopracovanie TODO, ktoré ostali
- 11552 – Zmazanie nepoužitých metód – v projekte sa nachádza mnoho metód, ktoré sa nepoužívajú, sú vhodné na vymazanie
- 11553 – Oprava pridávania hráča v Testframeworku – oprava znefunkčneného pridávania v rámci modulu
- 11554 – Doplnenie chýbajúcich analýz – nahratie chýbajúcich dokumentov na Wiki
- 11576 – Zápisnice – spracovanie zápisníc

2.9.3.2. Úlohy

Počas šprintu boli identifikované nasledujúce úlohy.

ID	Title	Assigned To	Priority	State	Completed Work
10611	implement the prediction module correctly	Bc. Filip Dian	2	Done	3
11275	Vyhodenie nepoužívaných súborov a konfigurácií IDE	Bc. Andrej Kútny	1	Done	2
11276	Vymazať nepoužité metódy 2	Bc. Adam Mikolášek	1	Done	2
11278	Doplniť chýbajúce vytvorené analýzy časť 1	Bc. Ľubomír Fischer	1	Done	4
11281	vypracovanie dokumentu - odovzdanie v 9 týždni	Bc. Kitti Nagyová	1	Done	2
11433	change conditions in method sequenceChceck	Bc. Ľubomír Fischer	2	Done	3
11459	Spracovanie návodu na nastavenie Maven projektu	Bc. Christopher Liebe	2	Done	1,5
11505	Oprava parametrov spúšťania agenta	Bc. Filip Dian	1	Done	6
11516	Doplnenie implementačných dokumentov na wiki 1	Bc. Zoltan Csengody	2	Done	2,5
11577	Spracovanie šprintu c. 8	Bc. Filip Dian	2	Done	1
11686	Vypracovanie Andrejovej časti	Bc. Andrej Kútny	2	Done	2

11687	Vypracovanie Kittinej časti	Bc. Kitti Nagyová	2	Done	2
11688	Vypracovanie Adinej časti	Bc. Adriana Beňovičová	2	Done	2
11689	Vypracovanie Zoltánovej časti	Bc. Zoltán Csengődy	2	Done	2
11690	Vypracovanie Ľubovej časti	Bc. Ľubomír Fischer	2	Done	2
11691	Vypracovanie Adamovej časti	Bc. Adam Mikolášek	2	Done	2
11692	Vypracovanie Filipovej časti	Bc. Filip Dian	2	Done	2
11709	Zápisnica 22.0	Bc. Zoltán Csengődy	2	Done	2
11710	Opis spúšťania agenta cez TF na Wiki	Bc. Filip Dian	2	Done	1
11711	Doplnenie implementačných dokumentov na wiki 2	Bc. Adriana Beňovičová	2	Done	3
11788	Vypracovanie jednoduchých TODO	Bc. Adam Mikolášek	2	Done	2

Tabuľka 27 Úlohy v šprinte č. 9

- 10611 – implement the prediction module correctly – vypracovanie TODO
- 11275 – Vyhodenie nepoužívaných súborov a konfigurácií IDE – zmazanie nepotrebných súborov
- 11276 – Vymazať nepoužité metódy 2 – vymazanie nepotrebných metód
- 11278 – Doplniť chýbajúce vytvorené analýzy časť 1 – nahratie chýbajúcich dokumentov na Wiki
- 11281 – Vypracovanie dokumentu – odovzdanie v 9. týždni – spracovanie dokumentu ku kontrolnému bodu
- 11433 – change conditions in method sequenceCheck – vypracovanie TODO
- 11459 – Spracovanie návodu na nastavenie Maven projektu – vypracovanie návodu, podľa ktorého nastavovať projekt po konvertovani projektu do Mavenu
- 11505 – Oprava parametrov spúšťania agenta – opravenie chybných parametrov príkazu na pridávanie hráča
- 11516 – Doplnenie implementačných dokumentov na Wiki 1 – nahratie chýbajúcich dokumentov
- 11577 – Spracovanie šprintu č. 8 – vypracovanie dokumentácie k šprintu

- 11686 – Vypracovanie Andrejovej časti – nahratie dokumentov do súhrnnej dokumentácie
- 11687 – Vypracovanie Kittinej časti - nahratie dokumentov do súhrnnej dokumentácie
- 11688 – Vypracovanie Adinej časti - nahratie dokumentov do súhrnnej dokumentácie
- 11689 – Vypracovanie Zoltánovej časti - nahratie dokumentov do súhrnnej dokumentácie
- 11690 – Vypracovanie Ľubovej časti - nahratie dokumentov do súhrnnej dokumentácie
- 11691 – Vypracovanie Adamovej časti - nahratie dokumentov do súhrnnej dokumentácie
- 11692 – Vypracovanie Filipovej časti - nahratie dokumentov do súhrnnej dokumentácie
- 11709 – Zápisnica 22.0 – spracovanie zápisnice
- 11710 – Opis spúšťania agenta cez TF na Wiki – nahratie dokumentácie opisujúcej spúšťanie agenta cez Testframework
- 11711 – Doplnenie implementačných dokumentov na Wiki 2 – nahratie chýbajúcich dokumentov
- 11788 – Vypracovanie jednoduchých TODO

2.9.4. Zhodnotenie

V šprinte sme opäť pokračovali na vypracovaní TODO a začali sme pracovať na refactoringu kódu. To znamená vymazanie nepoužívaných súborov a častí kódu. Okrem toho sme dopĺňali chýbajúce dokumenty na Wiki a nahrali všetky opisy dokončených implementácií do vlastnej kapitoly, pre budúce tímy. K 9. týždňu sa odovzdával dokument, ktorý všetky tieto veci obsahoval tiež. Všetka dôležitá práca v zdokumentovaniu sa v šprinte stihla, avšak znova ostali otvorené niektoré TODO.

2.9.4.1. Podiel práce členov tímu

Podľa vykazovania práce na taskoch bola vytvorená tabuľka podielov práce jednotlivých členov tímu. V celkovom počte hodín práce na taskoch je zarátaná aj práca strávená revidovaním a testovaním. Tasky, ktoré sa presunuli do ďalšieho šprintu zarátané nie sú.

Člen tímu	Formálne stretnutia	Neformálne stretnutia	Práca na taskoch
Bc. Filip Dian	6h	8h	13h
Bc. Christopher Liebe	6h	8h	1,5h
Bc. Andrej Kútny	6h	8h	4h
Bc. Ľubomir Fischer	6h	8h	9h

Bc. Adriana Beňovičová	6h	8h	5h
Bc. Kitti Nagyová	6h	8h	4h
Bc. Adam Mikolášek	6h	8h	6h
Bc. Zoltán Csengődy	6h	8h	6,5h

Tabuľka 28 Podiel práce členov tímu

2.10. Šprint č. 10

2.10.1. Základné informácie

Začiatok: 23.4.2019

Koniec: 10.5.2019

Tím:

Bc. Adriana Beňovičová

Bc. Zoltán Csengődy

Bc. Filip Dian

Bc. Ľubomír Fischer

Bc. Andrej Kútny

Bc. Adam Mikolášek

Bc. Kitti Nagyová

2.10.2. Retrospektíva

Dňa 6.5.2019 sme predbežne sumarizovali (ide o posledný šprint – trval 3 týždne) šprint č. 10. Urobili retrospektívu, počas ktorej sa každý člen vyjadril, čo sa mu páčilo a nepáčilo a určili si body, ktoré sa budeme v budúcnosti snažiť splniť. Pri bodoch sme aj určili, kto sa postará o jeho splnenie a v niektorých prípadoch aj vytvorili task.

2.10.2.1. Čo sa podarilo

- rozbeh písania častí finálneho dokumentu (Filip)
- komunikácia na slacku (Filip, Andrej)
- diskusia ohľadom obsahu finálneho dokumentu (Ľubomír)
- zvyšujúca sa kvalita dokumentov (Ľubomír)
- koordinácia Scrum Mastera
- rozvrhnutie prác na dokumentácii (Adriana, Adam, Andrej, Kitti)

2.10.2.2. Čo by sa mohlo zlepšiť

- opätovné rozbitie pridávania hráča (Filip)
- objavili sa nové chyby po posledných zmenách (Filip, Ľubomír, Adam, Andrej, Kitti)
- meškanie na stretnutia (Ľubomír, Andrej)
- nedorozumenie pri diskusii o vypracovaní časti dokumentu (Adriana)
- chybné nastavenie projektu, ktoré znemožňovalo prácu (Andrej)

2.10.2.3. Akčné body

- testovanie správania hráča pri každom commite (všetci)
- zhrnutie dôležitých bodov do dokumentu (Filip)
- dochvíľnosť (všetci)
- dobré opísaný task sa lepšie testuje (všetci)

2.10.3. Práca tímu

Na začiatku šprintu sme identifikovali prácu, ktorá by mala byť dokončená v rámci šprintu č. 10. Na základe vytvorených user stories sme zadali jednotlivé úlohy pre ich splnenie. Väčšina úloh bola vytvorená na začiatku šprintu a niektoré sa doplnili neskôr.

2.10.3.1. User stories

Do šprintu boli zaradené nasledujúce user stories.

ID	Title	Effort	State
10857	Oprava správania hráča (spartakiáda)	5	Done
11789	Zápisnice	1	Done
11791	Finálny dokument	2	Approved
11792	Finalizácia wiki	2	Done
11796	Príprava prezentácie	1	Approved
11839	Dokončenie TODO 2	5	Approved
11877	Odovzdanie produktu	1	Approved
11930	Oprava pridávania hráča	5	New

Tabuľka 29 User stories v šprinte č. 10

2.10.3.1.1. Opis user stories

- 10857 – Oprava správania hráča (spartakiáda) – implementácia zmysluplného pohybu v prípade, keď hráč nevie, čo má robiť
- 11789 – Zápisnice – spracovanie zápisníc
- 11791 – Finálny dokument – vypracovanie finálneho dokumentu
- 11792 – Finalizácia wiki – doplnenie Wiki o nové časti a uzavretie obsahu
- 11796 – Príprava prezentácie – spracovanie prezentácií k prezentovaniu finálnej podoby projektu
- 11839 – Dokončenie TODO 2 – dokončenie zvyšných TODO
- 11877 – Odovzdanie produktu – odovzdanie finálnej podoby projektu
- 11930 – Oprava pridávania hráča – Oprava parametrov spúšťania hráča v Testframeworku

2.10.3.2. Úlohy

Počas šprintu boli identifikované nasledujúce úlohy.

ID	Title	Assigned To	Priority	State	Completed Work
10609	try to create solution with reusable Singleton Factory	Bc. Filip Dian	2	Done	2
10839	implement calculation of opponent position		2	To Do	Future development
10841	A deviation has to be considered		2	To Do	Future development
10842	Implement computation with rotation agent to ball.		2	To Do	Future development
10852	accelerometer is relative to torsoPosition, not centerOfMass		2	To Do	Future development
11461	Zmena spartakiády na zmysluplný pohyb	Bc. Andrej Kútny	2	Done	6
11705	move to RoboCupLibrary	Bc. Filip Dian	2	Done	2
11706	referencie na iné kontrolery	Bc. Zoltán Csengődy	2	Done	2
11790	Zápisnica 23.0	Bc. Adam Mikolášek	2	Done	1
11793	Odkazy na TODO	Bc. Andrej Kútny	2	Done	1
11794	Vypracovanie tematického plánu	Bc. Ľubomír Fischer	2	Done	3
11795	Agent's goal is allways left, also during second period	Bc. Adam Mikolášek	2	Done	2
11797	Spracovanie prezentácie o projekte	Bc. Kitti Nagyová	2	In Progress	To be finished
11798	Spracovanie krátkej prezentácie	Bc. Kitti Nagyová	2	In Progress	To be finished
11858	Vytvorenie zdieľaného dokumentu	Bc. Kitti Nagyová	2	Done	1
11859	Opis aktuálneho stavu implementácie	Bc. Filip Dian	2	Done	3
11860	Opis stavu dokumentácie	Bc. Adriana Beňovičová	2	Done	2

11861	Opis dokumentácie testovania	Bc. Kitti Nagyová	2	Done	2
11862	Opis cieľov projektu	Bc. Ľubomír Fischer	2	Done	2
11863	Opis TODO	Bc. Adam Mikolášek	2	Done	2
11864	Kalmanov filter v kontexte plnenia cieľov	Bc. Adam Mikolášek	2	Done	1
11865	Dynamika pohybu v kontexte plnenia cieľov	Bc. Andrej Kútny	2	Done	1
11866	Euklidovské matice v kontexte plnenia cieľov	Bc. Kitti Nagyová	2	Done	1
11867	Multithreading v kontexte plnenia cieľov	Bc. Filip Dian	2	Done	2
11868	Optimalizácia Testframeworku v kontexte plnenia cieľov	Bc. Filip Dian	2	Done	1
11869	Otočenie osi x/y v kontexte plnenia cieľov	Bc. Ľubomír Fischer	2	Done	2
11870	Vyber pohybov v kontexte plnenia cieľov	Bc. Ľubomír Fischer	2	Done	2
11872	Oprava štruktúr v kóde v kontexte plnenia cieľov	Bc. Andrej Kútny	2	Done	2
11873	Opis chýbajúcej dokumentácie	Bc. Adriana Beňovičová	2	Done	2,5
11874	Opis budúcnosti projektu	Bc. Zoltán Csengődy	2	Done	3
11875	Zhodnotenie	Bc. Zoltán Csengődy	2	Done	1
11876	Zápisnica 24.0	Bc. Kitti Nagyová	2	Done	1
11878	Uloženie statickej stránky a zdrojového kódu na CD	Bc. Filip Dian	2	To Do	To be finished
11927	Spracovanie šprintu c. 9	Bc. Filip Dian	2	Done	2
11931	Oprava pridávania hráča v Testframeworku	Bc. Filip Dian	2	To Do	To be finished
11932	Globálna retrospektíva	Bc. Filip Dian	2	Done	2

11933	Doplnenie dokumentov za šprint 9, 10 - Filip	Bc. Filip Dian	2	Done	0.5
11937	Zápisnica 25.0	Bc. Andrej Kútny	2	Done	1
11963	Doplnenie dokumentov za šprint 9, 10 - Adriana	Bc. Adriana Beňovičová	2	Done	0.5
11964	Doplnenie dokumentov za šprint 9, 10 - Ľubomír	Bc. Ľubomír Fischer	2	Done	0.5
11965	Doplnenie dokumentov za šprint 9, 10 - Adam	Bc. Adam Mikolášek	2	Done	0.5
11966	Doplnenie dokumentov za šprint 9, 10 - Andrej	Bc. Andrej Kútny	2	Done	0.5
11967	Doplnenie dokumentov za šprint 9, 10 - Zoltan	Bc. Zoltán Csengődy	2	Done	1
11968	Doplnenie dokumentov za šprint 9, 10 - Kitti	Bc. Kitti Nagyová	2	Done	1
11969	Spracovanie šprintu c. 10	Bc. Filip Dian	2	Done	2

Tabuľka 30 Úlohy v šprinte č. 10

2.10.3.2.1. Opis úloh

- 10609 – try to create solution with reusable Singleton Factory – vypracovanie TODO
- 10839 – implement calculation of opponent position - vypracovanie TODO
- 10841 – A deviation has to be considered - vypracovanie TODO
- 10842 – Implement computation with rotation agent to ball - vypracovanie TODO
- 10852 – accelerometer is relative to torsoPosition, not centerOfMass - vypracovanie TODO
- 11461 – Zmena spartakiády na zmysluplný pohyb – implementácia zmysluplného pohybu v prípade, keď hráč nevie kde sa nachádza
- 11705 – move to RoboCupLibrary - vypracovanie TODO
- 11706 – referencie na iné kontrolery - vypracovanie TODO
- 11790 – Zápisnica 23.0 – vypracovanie zápisnice
- 11793 – Odkazy na TODO – doplnenie odkazov na Wiki
- 11794 – Vypracovanie tematického plánu – spracovanie plánu obsahu finálneho dokumentu
- 11795 – Agents goal is allways left, also during second period - vypracovanie TODO
- 11797 – Spracovanie prezentácie o projekte – spracovanie prezentácie o projekte
- 11798 – Spracovanie krátkej prezentácie – spracovanie krátkej prezentácie

- 11858 – Vytvorenie zdieľaného dokumentu – vytvorenie a sprístupnenie odkazu na finálny dokument
- 11859 – Opis aktuálneho stavu implementácie – vypracovanie časti finálneho dokumentu
- 11860 – Opis stavu dokumentácie - vypracovanie časti finálneho dokumentu
- 11861 – Opis dokumentácie testovania - vypracovanie časti finálneho dokumentu
- 11862 – Opis cieľov projektu - vypracovanie časti finálneho dokumentu
- 11863 – Opis TODO - vypracovanie časti finálneho dokumentu
- 11864 – Kalmanov filter v kontexte plnenia cieľov - vypracovanie časti finálneho dokumentu
- 11865 – Dynamika pohybu v kontexte plnenia cieľov - vypracovanie časti finálneho dokumentu
- 11866 – Euklidovské matice v kontexte plnenia cieľov - vypracovanie časti finálneho dokumentu
- 11867 – Multithreading v kontexte plnenia cieľov - vypracovanie časti finálneho dokumentu
- 11868 – Optimalizácia Testframeworku v kontexte plnenia cieľov - vypracovanie časti finálneho dokumentu
- 11869 – Otočenie osí x/y v kontexte plnenia cieľov - vypracovanie časti finálneho dokumentu
- 11870 – Výber pohybov v kontexte plnenia cieľov - vypracovanie časti finálneho dokumentu
- 11871 – Testovanie nových knižníc v kontexte plnenia cieľov - vypracovanie časti finálneho dokumentu
- 11872 – Oprava štruktúr v kóde v kontexte plnenia cieľov - vypracovanie časti finálneho dokumentu
- 11873 – Opis chýbajúcej dokumentácie - vypracovanie časti finálneho dokumentu
- 11874 – Opis budúcnosti projektu - vypracovanie časti finálneho dokumentu
- 11875 – Zhodnotenie - vypracovanie časti finálneho dokumentu
- 11876 – Zápisnica 24.0 – vypracovanie zápisnice
- 11878 – Uloženie statickej stránky a zdrojového kódu na CD – odovzdanie zdrojových kódov a Wiki na CD
- 11927 – Spracovanie šprintu č. 9 – spracovanie sumarizácie predošlého šprintu
- 11931 – Oprava pridávania hráča v Testframeworku – opravenie parametrov spúšťania hráča cez Testframework
- 11932 – Globálna retrospektíva – spracovanie retrospektívy za celý projekt
- 11933 – Doplnenie dokumentov za šprint 9, 10 – nahratie nových dokumentov na Wiki
- 11937 – Zápisnica 25.0 – spracovanie zápisnice
- 11963 – Doplnenie dokumentov za šprint 9, 10 - Adriana - nahratie nových dokumentov na Wiki
- 11964 – Doplnenie dokumentov za šprint 9, 10 – Ľubomír - nahratie nových dokumentov na Wiki

- 11695 – Doplnenie dokumentov za šprint 9, 10 – Adam - nahratie nových dokumentov na Wiki
- 11966 – Doplnenie dokumentov za šprint 9, 10 – Andrej - nahratie nových dokumentov na Wiki
- 11967 – Doplnenie dokumentov za šprint 9, 10 – Zoltán - nahratie nových dokumentov na Wiki
- 11968 – Doplnenie dokumentov za šprint 9, 10 – Kitti - nahratie nových dokumentov na Wiki
- 11969 – Spracovanie šprintu č. 10 – spracovanie (predbežnej) sumarizácie aktuálneho šprintu

2.10.4. Zhodnotenie

V šprinte sme sa najprv snažili dokončiť otvorené TODO úlohy a ďalej sme sa sústredili na vypracovávanie finálneho dokumentu. Jednotlivé časti sme rozdelili medzi členov tímu. Potom sme súčasne pracovali na finalizácií dokumentu. TODO, na ktorých sa ešte nezačalo pracovať, už rozpracované nebudú. Oprava pridávania agenta ostáva ako práca po oficiálnom ukončení projektu a odovzdaní finálnej dokumentácie. Tiež ešte bude finalizované odovzdávanie a prezentovanie projektu.

2.10.4.1. Podiel práce členov tímu

Podľa vykazovania práce na taskoch bola vytvorená tabuľka podielov práce jednotlivých členov tímu. V celkovom počte hodín práce na taskoch je zarátaná aj práca strávená revidovaním a testovaním. Práca na finálnom dokumente tu zarátaná nie je. Zahŕňa to zmeny na základe review a formátovanie svojich častí v dokumente.

Člen tímu	Formálne stretnutia	Neformálne stretnutia	Práca na taskoch
Bc. Filip Dian	6h	8h	16,5h
Bc. Andrej Kútny	6h	8h	11,5h
Bc. Ľubomir Fischer	6h	8h	9h
Bc. Adriana Beňovičová	6h	8h	5h
Bc. Kitti Nagyová	6h	8h	6h
Bc. Adam Mikolášek	6h	8h	6,5h

Bc. Zoltán Csengődy	6h	8h	7h

Tabuľka 31 Podiel práce členov tímu

2.11. Globálna retrospektíva

2.11.1. Čo sa podarilo

- naučenie cyklu agilného vývoja (Ľubomír, Adam, Filip)
- očividné zlepšenie tímovej práce (Ľubomír)
- vďaka dôslednej (byrokratickej) príprave sa zlepšovala kvalita implementácií a prehľadnosť dokumentovania (Ľubomír)
- pri konci projektu sme sa zhodli na cieli a výslednom výstupe (Adriana)
- nápomocná práca Scrum Mastera (Adriana, Adam)
- lepší tímový duch v druhej polovici projektu (Adriana)
- získanie prehľadu v projekte (Andrej)
- dodržiavanie dôležitých mechaník scrumu (Andrej)
- dobrý tímový duch (Andrej, Adam, Filip)
- každý v tíme mal nejaký odborný prínos (Andrej)
- komunikácia aj mimo stretnutí (Andrej)
- odchod člena tímu neovplyvnil ďalší vývoj (Zoltán)
- voľnosť v tíme (Zoltán)
- aj napriek absencii úvodu sme sa dobre zorientovali v problematike (Zoltán, Filip)
- vytvorenie kvalitných dokumentácií pre ďalšie tímy (Zoltán)
- poupratovanie projektu (Zoltán)
- príjemné stretnutia (Kitti)
- viditeľnosť výsledok práce v projekte (Kitti)
- dobrá spolupráca a komunikácia všetkých členov tímu (Kitti)
- úprava procesov podľa potrieb tímu (Adam, Filip)
- rozhodnutie viac sprehľadniť kód miesto veľkých implementácií (Adam)
- prvé stretnutie s členom predošlého tímu (Adam)
- spracovanie celkovej dokumentácie projektu (Filip)

2.11.2. Čo by sa mohlo zlepšiť

- malá spolupráca s product ownerom projektu (Ľubomír, Zoltán)
- ťažko hmatateľný finálny cieľ projektu negatívne vplýval na motiváciu jednotlivých členov tímu (Ľubomír)
- projekt je veľmi chaotický a neucelený, neaktualizovaný (Adriana, Kitti)

- nestanovenie cieľa práce na tímovom projekte (Adriana)
- nerozhodnosť jedného člena tímu ohľadom pokračovania v projekte (Adriana)
- slabý úvod do projektu (Andrej)
- celý backlog tvoril iba tím (Andrej)
- občasné nedorozumenia pri plnení cieľov, odmietanie spracovania niektorých úloh (Zoltán, Filip)
- rozhodnutie nepokračovať na niektorých zložitých úlohách (Zoltán)
- zdĺhavá (aj keď potrebná) analýza oblasti projektu (Kitti, Adam)
- málo informácií od predošlých tímov (Kitti)
- obtiažne testovanie niektorých úloh (Kitti)
- malé pokrytie unit testami (Kitti)
- odchod člena tímu (Kitti, Adam)
- časté vytváranie práce bez product ownera (Adam)
- menej dokončených implementácií ako sa očakávalo (Adam, Filip)
- časté meškanie dokončenia úloh v rámci šprintov (Filip)
- nedodržiavanie niektorých stanovených pravidiel, metodík (Filip)
- meškanie na pondelkové stretnutia (Filip)

2.11.3. Akčné body

- precíznosť menšieho množstva implementácií má väčší zmysel pre projekt ako bezhlavá implementácia väčších funkcionalít
- nutnosť skorého definovania cieľa a priebežnej konzultácie s product ownerom
- dôsledné spracovanie metodík a ich dodržiavanie prináša sprehľadnenie práce a zjednodušenie nadviazania ďalšími členmi
- priebežné rozpracovanie úloh v šprinte
- naplánovanie stretnutí v rozumnom čase

3. Aktuálny stav projektu

3.1. Stav implementácie

V projekte sa nachádza niekoľko „implementačných“ celkov, ktoré boli dlhodobo rozvíjané v priebehu rokov tímami študentov. Niektoré z nich sú stále rozpracované resp. sa neustále vylepšujú a ostatné bolo možné uzavrieť.

3.1.1. Logovanie a jeho optimalizácia

V projekte sa používa na logovanie klasicky trieda java.util.logging.Logger. Logovanie má niekoľko typov v závislosti od logovanej informácie. Pridávanie typov logovania je možné cez GUI. Takisto je logovanie rozdelené aj na základe závažnosti správy. Úroveň sa taktiež nastavuje v GUI.

Pri analyzovaní časovej náročnosti kódu bolo zistené, že logovanie nie je na niektorých miestach správne použité, a správa sa vytvára aj vtedy, keď logovanie nie je zapnuté. To v niektorých prípadoch znamenalo veľkú náročnosť, ktorá bola odhalená pomocou programu Yourkit. Na základe analýzy potom bolo logovanie upravené tak, aby sa celá správa vytvárala iba vtedy, keď je potrebné (daný typ logovania je zapnutý).

3.1.2. Zisťovanie polohy hráča

Počas rokov sa táto oblasť projektu neustále vyvíjala, a predposledným tímom bolo navrhnuté, že sa vylepší presnosť odhadovania polohy na základe identifikácie vzorov čiar resp. ich priesečníkov, ktoré hráč vidí. Momentálne odhadovanie fungovalo len na základe histórie, údajov z akcelerometra a fixných objektov, ktoré vidí a ich relatívna poloha je oznamovaná serverom. Boli implementované funkcie na identifikáciu priesečníkov v tvare +, T a bodu bránkoviska. Tie boli potom použité na upresnenie polohy, keď hráč vidí jeden z týchto vzorov. Implementácie boli otestované a sú považované za funkčné. Pre testovanie bol aj vytvorený nový typ logovania (typ POINTS), ktorý sa dá zapnúť v GUI. V implementácií ale nie sú zahrnuté prípady (kvôli použitej logike), napr. keď je hráč presne v strede medzi bodmi, ktoré sú označené ako TODO. Takisto nie sú zahrnuté odchýlky. Bolo by dobré sa týmto veciam v budúcnosti venovať.

3.1.3. Prevrátenie osí X/Y

V jadre spracovávania súradníc sú v niektorých vektorových výpočtoch zamenené osi X a Y. Tým pádom všetka logika nad týmito výpočtami (určovanie polohy, pohyb apod.) je postavená

s tým, že sú tieto súradnice prehodené. Preto nie je priame prehodenie na najnižšej úrovni triviálnym riešením. Tím sa touto problematikou zaoberal, avšak zisťovanie, kde všade sú súradnice prehodené zabralo príliš veľa času a z úlohy sa upustilo. Opravenie osí vyžaduje mnoho času na analýzu a odporúčame, aby si tím, ktorý bude problematiku riešiť, vyhradil dostatok času.

3.1.4. Zero Moment Point a Kalmanov filter

Niektorým z predošlých tímov bol vytvorený opis a čiastočná implementácia techniky ZMP. Tá sa používa na stabilizovanie objektu pri pohybe. Aktuálny stav však na začiatku pracovania na projekte nebol dostatočný a funkčný a tak sa tím zaoberal aj analýzou tejto oblasti. Analýza sa nachádza v samostatnej kapitole ďalej v dokumente.

Kalmanov filter bol taktiež analyzovaný a rozpracovaný jedným z predošlých tímov a bolo potrebné na ňom ďalej pracovať. V projekte už používaný je, ale nebolo isté, či funguje správne. Tiež bolo navrhnuté, aby sa filter optimalizoval a boli spomenuté miesta, kde by sa dal ďalej použiť. Problematike sa venuje podrobnejšie samostatná kapitola ďalej v dokumente.

3.1.5. Práca na TODO

Predošlými tímami bolo identifikovaných mnoho TODO, ktoré nestihli alebo nevedeli spracovať. Snaha bola počet TODO čo najviac zredukovať. Väčšinou išlo o menšie zmeny, ktoré sa podarilo implementovať bez nejakých problémov. Niektoré zložitejšie veci sa dokončili a niektoré sa zavrhli, pretože nemali zmysel, alebo sa jednoducho neoplatili implementovať. Tím vyčistil veľkú časť TODO. Jednotlivé implementácie sú opísané v samostatných kapitolách.

3.1.6. Iné implementácie

Počas práce na projekte sme sami identifikovali niekoľko problémov resp. vylepšení, na ktoré by bolo vhodné sa zamerať. Ide hlavne o optimalizácie a doplnenie funkcií, ktoré by pomohli pri predovšetkým pri testovaní (hlavne z oblasti TestFrameworku). Tieto veci sú na základe backlogu opísané v samostatných kapitolách.

3.2. Dokumentácia

3.2.1. WIKI

V rámci Tímového projektu sme pracovali s WIKI pre získanie prehľadu ako s projektom Robotickú futbal pracovať. Prvotne bolo dôležité samotný projekt rozbehať na našich osobných

PC, s čím nám pomohol minuloročný žiak, ktorý absolvoval predmet Tímový projekt s touto témou. Tento proces je opísaný aj na wiki no vzhľadom k zložitosti a potrebným nástrojom pre rozbehanie projektu, bolo pre nás ľahšie a rýchlejšie, keď nám s touto fázou pomohol už absolvent predmetu Tímový projekt.
Následné zahĺbenie do problematiky, spájanie a pochopenie logiky programu bola úzko späté so spoluprácou WIKI, kde sme dúfali, že nájdeme všetky odpovede na otázky, ktoré nám počas práce na projekte vznikali. Pochopili sme však, že táto dokumentácia nebola aktualizovaná a ani podrobná do takej miery, akú by sme pre pochopenie projektu a jeho následné vylepšovanie potrebovali. Rovnako sa vyskytovalo veľmi veľa nepochopených a nevysvetlených častí kódu, komentárov pre odporúčané dopracovanie programu, ktoré však nemali žiadny bližší a hlbší popis či odôvodnenie. Orientácia v rámci projektu bola veľmi ťažká a zdĺhavá, množstvo informácii sme spracovali v rámci tímovej hypotézy či predpokladu, ako by to mohlo fungovať.

Práve zo spomínaných dôvodov, sme sa rozhodli v rámci tímu, aktualizovať a zapracovať všetky dokumentácie, implementácie a informácie, ktoré sa nám podarilo vytvoriť a zoskupiť na WIKI, pre ďalšie tímy. Aktualizovali sme všetky informácie, na ktoré sme prišli, ako aj pridali nové funkcie, o ktoré sme projekt vylepšili. Zároveň, pre ucelenie vypracovaných dokumentov sme na WIKI vytvorili kapitolu „9.implementacia“, ktorá obsahuje podkapitolu „tp_2018_2019“, kde sme združili všetky vypracované dokumentácie za tím WAWOT, počas celého roku. Veríme, že aj táto skutočnosť, že sme uviedli presné dokumentácie, na čom sme počas predmetu Tímový projekt pracovali, pomôže lepšiemu a rýchlejšiemu zahĺbeniu do problematiky nasledujúcemu tímu. Rovnako veríme, že pri vznikajúcich problémoch, bude budúci tím vedieť lepšie dohľadať odpovede na vzniknuté otázky, už len z dôvodu, že bude vedieť či obdobný problém bol riešený našim tímom alebo je to problém, ktorý pramení od starších tímov.

3.2.2. Analýza a implementácia

Začiatok projektu, z dôvodu absencie užitočných dokumentov, bol zameraný na analyzovanie aktuálneho stavu oblastí riešených v projekte. Analýzy potom boli spracované v dokumentoch, podľa ktorých sme následne naštartovali vývoj. Ide hlavne o oblasti, ktoré sa spomínajú vyššie. Po ich dôkladnom analyzovaní sme sa pustili do implementácie riešení, či už nami navrhnutými alebo ešte predošlými tímami, ktoré sme taktiež samostatne dokumentovali. Všetky spracované analýzy aj implementácie sme postupne nahrávali na Wiki, a sú súčasťou aj tohto dokumentu.

3.2.3. Testovanie

Počas semestra sme vytvorili metodiku testovania, ktorej sme sa držali pri testovaní každej jednej úlohy, kde bolo overenie správnosti nevyhnutné. Metodika testovania hovorí o správnosti vykonania testu, o testovacích scenároch a o ďalšom postupe na úlohe. Každý jeden vykonaný test má vytvorený dokument testovania, ktorý obsahuje všetky nevyhnutné časti opísané v metodike testovania. Ako negatívum hodnotím nízke pokrytie projektu jednotkovými testami, ktoré by sa mohlo napraviť v budúcej práci na projekte. Taktiež súčasné jednotkové testy nie sú na 100% úspešné, bolo by treba pozrieť sa na pôvod problému. Prepracovanie TestFrameworku by tiež pomohlo pri testovaní implementácií. Vysokú optimalizáciu projektu a testovanie optimalizácie sme dosiahli vďaka analyzátoru „yourkit“. Bolo by vhodné vymyslieť aj správny a jednotný spôsob manuálneho testovania úloh, ktoré nie je možné otestovať jednotkovými testami.

4. Ciele projektu

4.1. Ciele projektu a ich naplnenie

Základom úspešného napredovania v projekte je stanovenie si jasných a dosiahnuteľných cieľov. Ciele, ktoré sme si stanovili na začiatku projektu vychádzali jednak z pohľadu product ownera, ktorý v úvodných diskusiách špecifikoval kritické oblasti a predstavy o fungovaní produktu. Druhým zdrojom úvodných cieľov projektu bolo odovzdanie a finálna dokumentácia tímu, ktorý pracoval na robotickom hráčovi pred nami.

Tieto ciele mali za úlohu hneď od začiatku stanoviť smerovanie snahy a práce celého tímu.

Ciele zimného semestra:

1.
    a. Cieľ: veľká časť dostupných zdrojov je spotrebovaná hráčom pri získavaní informácii zo servera
    b. Riešenie: analýza zdrojového kódu s použitím nástroja YourKit Profiler. Analýzou bolo zistených viacero neoptimálnych implementácií z pohľadu časovej náročnosti, najvýznamnejšia z nich bola logovanie. Výrazne časovo náročné úlohy boli optimalizované, čo malo za následok zníženie výpočtovej náročnosti programu, čím sme umožnili lepšie fungovanie hráča aj na slabších strojoch.
2.
    a. Cieľ: zlá orientácia hráča v rámci ihriska, pri náklonoch hlavy dochádza k chybnej identifikácií polohy
    b. Riešenie: implementované funkcie na identifikáciu priesečníkov čiar na ihrisku v tvare +, T a bodu bránkoviska. Použitie tohto spôsobu lokalizácie pre upresnenie polohy, keď hráč vidí jeden z týchto vzorov zlepšilo rozhodovacie schopnosti hráča – rýchlejšie a efektívnejšie rozhodovanie o pohybe za loptou. Implementácie prešli testami a boli použité v programe. Pre testovanie bol vytvorený nový typ logovania (typ POINTS).
3.
    a. Cieľ: prehodené osi X a Y počas spracovania obrazových informácii zo servera (segment SEE v S-výrazoch)

b. Riešenie: analýzou triedy Vector3D sme priblížili miesto, kde sa vykonáva výpočet súradníc a základné operácie s nimi. Túto analýzu je však ďalej potrebné rozšíriť o matematickú logiku, na ktorej sú metódy pracujúce so súradnicami postavené. Po vypracovaní podrobnej dokumentácie je možné skúšať prehodiť osi X, Y korektným spôsobom.

Ciele letného semestra:

1.
    a. Cieľ: vylepšenie správania simulovaného hráča k dosiahnutiu lepších výsledkov
    b. Riešenie: tento všeobecne stanovený cieľ bol naplnený predovšetkým zapracovaním rôznych menších implementačných, či optimalizačných úloh, opísaných nižšie v projekte, predovšetkým plnením cieľov stanovených v zimnom semestri a riešením úloh TODO v kóde
2.
    a. Cieľ: sprehľadnenie fungovania implementácie – analýza existujúcich funkcionalít
    b. Riešenie: sprehľadnenie bolo uchopené najmä z pohľadu lepšieho dokumentovania zmien, zvýšeného dôrazu na testovanie zmien a vzájomného ovplyvňovania funkcionalít po vykonaných zmenách. Analýza priniesla zase prehľad nad už existujúcimi časťami projektu. Je potrebné ju však neustále prehlbovať, nakoľko zdroje k implementáciám predchádzajúcich tímov nie sú prehľadné a v mnohých prípadoch ani k dispozícii.
3.
    a. Cieľ: opravenie zistených nedostatkov počas letného semestra
    b. Riešenie: vytváranie splniteľných a jasne špecifikovaných úloh, ich systematické plnenie s efektívnou komunikáciou v rámci tímu.

Ciele identifikované v rámci práce počas letného semestra:

1. implementácia identifikovania vzorov čiar pre prípady keď je hráč presne v strede medzi bodmi, ktoré sú označené ako TODO. Takisto nie sú zahrnuté odchýlky.
2. opravenie osí vyžaduje množstvo času na analýzu, odporúčame, aby si tím, ktorý bude problematiku riešiť, vyhradil dostatok času
3. optimalizácia a použitie kalmanovho filtra v projekte
4. dokončenie TODO vybraných z projektu

5. optimalizácie a doplnenie funkcií pre testovanie (najmä TestFrameworku)
6. udržiavanie aktuálneho stavu dokumentácie implementácií na wiki

Ciele, ktoré navrhujeme pre ďalšie pokračovanie na projekte sme sa snažili čo možno najpodrobnejšie opísať aj v jednotlivých kapitolách a analýzach. Okrem iného sme všetky implementačné zmeny podrobne zdokumentovali na wiki a otestovali aj voči spätnému ovplyvneniu funkcionalít po ich nasadení. Našim výsledok je komplexná generálna aktualizácia a sprehľadnenie wiki.

Za nesmierne dôležitý krok v rámci budúceho plnenia cieľov vidíme dopĺňanie a udržiavanie aktuálnosti časti „9. implementacia“, v ktorej je potrebné dokumentovať predovšetkým implementačné zmeny. Tento krok v budúcnosti uľahčí prípadné analýzy nových nápadov a optimalizácie existujúcich častí robotického hráča futbalu Jima. Vo výsledku by mala byť wiki primárnym zdrojom informácií pre výskum a implementáciu nových funkcií.

4.2. Problémy súčasnej implementácie

4.2.1. Optimalizácia Testframeworku

Logika Testframeworku bola predošlými tímami zdokumentovaná iba minimálne. Funkcia a celkový zmysel jednotlivých častí preto nikdy nebola dostatočne vysvetlená. Počas práce sa ale ukázali niekoľké nedostatky, ktoré znepríjemňovali testovanie a debugovanie implementovaných častí.

V prvom rade je to absencia vypisovania logov, ktoré je súčasťou štandardného behu programu (Jima). Bez logovania nebolo možné testovať niektoré implementované súčasti pri potrebe funkcionality Testframeworku (polohovanie hráča a zobrazovanie čiar). Túto časť by bolo vhodné pridať ako nový tab v GUI.

Ďalšou problematickou časťou je logika pridávania hráča. To spočíva v spustení nového Java procesu pomocou definovaného príkazu s parametrami, ktoré sú fixne uložené v súbore sk/fiit/testframework/init/default.properties. Pri manipulácií s knižnicami pri konvertovaní projektu do Mavenu bolo pridávanie znefunkčnené. Kvôli absencií dokumentácie bolo pri vyskytnutom probléme ťažké dohľadať príčinu, čo spôsobilo, že oprava trvala niekoľko šprintov. Problém je opísaný v rámci wiki, avšak by bolo dobré premyslieť danú logiku a implementovať ju iným spôsobom, minimálne dynamicky skladať parametre spustenia.

Celková absencia zdokumentovania daného modulu spôsobila, že tím poriadne nevedel, načo a ako sa dá Testframework použiť. Navrhujeme preto sa hlbšie pozrieť do GUI a zdokumentovať jednotlivé funkcie.

4.2.2. Multithreading

V záujme optimalizácie je komunikácia so serverom oddelená do samostatného vlákna, ktoré sa spúšťa spolu s hráčom. Riešenie je navrhnuté tak, že je možné spustiť viac komunikačných vlákien naraz. Pri inicializácii programu s viac než jedným vláknom však čelíme problému, ktorý spôsobuje, že sú prostriedky vždy prideľované len jednému vláknu a ostatné "hladujú". Maximálny počet je preto nastavený na hodnotu 1. Napriek tomu bol chod programu zrýchlený o približne 9%.

Správy zo servera sú spracované metódami triedy Parser.java. Momentálne v nej je implementovaný návrhový vzor Singleton, aby nebolo potrebné zakaždým vytvárať nové inštancie. Korektné spracovanie a rozposlanie správy je kritické pre správny chod programu a z toho dôvodu je umiestnené v zámku (implementované ako ReentrantLock).

4.2.3. Otočenie osí x/y

Jedným z dlhodobých cieľov identifikovaných ešte predchádzajúcim tímom BAREKO bolo otočenie osí X a Y v projekte. Cieľu sa venovali commity acd9c65 resp. 1bc3ebb. O riešenie sa s čiastočným úspechom pokúšali v branchi TP_2017_fix_invertovanych_suradnic. Daná úloha sa nepodarila dokončiť vďaka neexistencií analýzy tohto problému.

Náš tím WAWOT vykonal viacero analýz tried a metód, kde potenciálna inverzia nastala. Snaha bola nájsť miesta vo vyšších úrovniach spracovania súradnicového systému, kde by boli osi očividne prehodené. Nájdené použitia na základe "konštruktorov" Vector3D.cartesian a Vector3D.spherical boli skontrolované a prípadné opačné použitia osí boli prehodené a otestované. Ďalej boli otočené metódy spherical, calculateSpherical, calculateCartesian, rotateOverX, rotateOverY a rotateOverZ. Ani jedna z týchto zmien nepriniesla očakávaný výsledok. Zhodnotili sme, že vzhľadom na rozpracované úlohy v našom projekte by ďalšie hodiny nezodpovedali vynaloženému úsiliu.

Pre ďalší tím odporúčame od základov analyzovať predovšetkým celú triedu Vector3D, a prekopať súčasné výpočty súradníc, operácie nad nimi a skontrolovať ich matematickú reprezentáciu. V aktuálnom stave však inverzia súradníc nehrá veľkú rolu pri pohyboch, rozhodovaní a orientácií robotického hráča.

4.2.4. Euklidovské matice

Tieto matice sa používajú pri vektorových výpočtoch pri transformáciách priestorových útvarov so zachovaním veľkosti. V projekte sú využívané len na výpočet orientácie hráča a zhodli sme sa, že by bolo efektívne ich využiť aj inde. Takisto sa zrejme v implementáciach nachádzajú chyby. Všetky transformácie sú opísane vo vlastnej kapitole.

4.2.5. Kalmanov Filter

Kalmanov filter je nástroj, ktorý predstavuje výpočtovo účinný prostriedok k odhadu stavu procesu. V projekte sa kalmanov filter využíva na upravovanie súradníc polohy lopty a pevných bodov, ktoré prichádzajú zo servera. Tieto súradnice prichádzajúce zo servera nie sú úplne presnéa prichádzajú hráčovi s istou chybou – so šumom. Kalmanov filter teda slúži na redukovanie tohto šumu aby hráč mohol presnejšie určiť polohu lopty a pevných bodov. Na úpravu polohy lopty sa využíva novší kalmaov filter využívajúci matice nachádzajúci sa v triede KalmanReal, pričom na úpravu polôh pevných bodov sa používa starší menej efektívny kalmanov filter KalmanForVector. V rámci práce na projekte sme kalmanov filter analyzovali, avšak kvôli nižsej priorite úlohy sme sa nedostali k implementácií nového kalmanovho filtra aj pre pevné body zo serveru. Preto by bolo vhodné túto implementáciu dokončiť.

4.2.6. Dynamika pohybu

Aktuálne riešenie rozdelenia pohybu na High skills a Low skills zadáva robotovi presné pohyby, ktoré má v danej situácii vykonať. Nastáva problém s tým, keď chceme vykonať pohyb v inom uhle, rozpätí, dĺžke ako má preddefinované. Vo veľa situáciach by sa vyplatilo práve tieto lowskills, ktoré sú popisané v statických xml súboroch nahradiť dynamickým pohybom, ktorý by napomohol dané pohyby viac prispôsobiť situácii.

Ako príklad môže byť otáčanie hlavy o 120 stupňov v prípade straty orientácie Jima. Predstavme si, že Jimovi stačí na určenie polohy posunúť hlavu o 5 stupňov a späť. Namiesto týchto 10 stupňov vykoná 240 stupňový pohyb aj napriek tomu, že už spočiatku vedel vypočítať svoju polohu. Pohyb hlavy nie je až tak závažný problém, ktorý by dynamika pohybu kĺbov vyriešila. Najväčší vplyv by to mohlo mať na pohyb.

Metóda Zero Moment Point (ZMP) vie veľmi dobre pracovať s Center Of Mass (COM) a prispôsobovať pohyb polohe ťažiska v robotovi. Táto dynamickosť pohybu pri Jimovi vedela priniesť rýchlejšiu a stabilnejšiu chôdzu.

4.2.7. Výber pohybov

Zlá nadväznosť pohybov a často ich nelogický výber v trajektórií boli jedným z veľmi jasne viditeľných problémov v behu robotického hráča Jima. Jim vypočíta trajektóriu v triede TrajectoryPlanner a následne overí možnosť nadväznosti jednotlivých pohybov v triede MoveValidator.

Prvotný problém, ktorý sme riešili bolo dopracovanie TODO v metóde sequenceCheck. Veľmi všeobecný popis hovoriaci o „provizórnom riešení nadväznosti pohybov“ nás prinútil vystavať analýzu týchto tried pre pochopenie celého procesu plánovania pohybov. Boli identifikované možné optimalizácie podmienok pri zlej nadväznosti naplánovaných pohybov trajektórie v metóde sequenceCheck, ktoré sme následne implementovali a otestovali. Konkrétne sa jednalo o prerušenie trajektórie v prípade pádu Jima, v prípade príliš veľkej odchýlky v predpokladaných súradniciach ďalšieho pohybu a aktuálnej pozície agenta a v prípade nemožnosti vykonať naplánovaný pohyb z dôvodu nevyhovujúcej polohy kĺbov robota. Optimalizácia sa týkala podmienky, keď robot spadol a je popísaná nižšie v ďalších kapitolách.

Analýza triedy TrajectoryPlanner tiež odhalila neefektívnu podmienku vo vykonávaní trajektórie, kedy robotický hráč zastavil svoj pohyb v ¾ trasy. Túto podmienku sme zrušili a robotický hráč sa tak dostane k cieľu rýchlejšie a bez zastavenia.

Výber pohybov však stále nie je dokonalý, Jim mnohokrát zvolí nevhodný pohyb, prípadne vypočítaná trajektória je príliš nepresná. Navrhujeme pokračovať v analýze triedy TrajectoryPlanner aj v kontexte ďalších metód a tried, ktoré sú s ňou prepojené až po vykonávanie samotných pohybov. Obzvlášť pri reálnom modeli hry futbalu je nevyhnutné, aby hráč reagoval správne a jeho trajektória bola čo najpresnejšia.

4.2.8. Použité štruktúry a kvalita kódu

Počas práce na úlohe so zmenou údajovej štruktúry pre udržiavanie situácii sme si všimli, že vo väčšine prípadov bol bez rozmýšľania použitý ArrayList. ArrayList má určite veľa výhod, no rozhodnúť sa ho použiť všade (a bez predpokladanej veľkosti) je dosť nešťastným riešením. Nielen že toto riešenie znižuje efektivitu výpočtu, no je taktiež pamäťovo náročnejšie ako v prípade vhodne zvolenej údajovej štruktúry.

Určite odporúčam prejsť jednotlivé časti kódu, ktoré udržiavajú nejaké dáta, vyskúšať debug mód a zamyslieť sa, či by nebolo vhodnejšie použiť inú štruktúru ako sa v kóde aktuálne nachádza.

Ďaľšou, veľmi problematickou časťou programu je, že kvalita kódu nie je vždy ideálna. V spojení s údajovými štruktúrami sme sa stretli počas prehliadky kódu napríklad so situáciou, kedy sa prehľadával for cyklom (nie foreach) LinkedList. Toto riešenie by bolo pri veľkej údajovej štruktúre veľmi výpočtovo náročné a ovplyvnilo by správanie Jima.

Okrem nesprávne použitých algoritmov na prehľadávanie štruktúr je v kóde veľa nekvalitných častí kódu, ktoré by bolo lepšie naprogramovať inak, efektívnejšie. Jeden z problémov projektu na ktorom pracuje viacero tímov je práve to, že review kódu sa robí všelijak. Počas prehliadky kódu je jasné, že niektoré funkcionality neprešli žiadnou kontrolou, respektíve kontrolou s veľmi nízkymi nárokmi.

4.3. Identifikovaná práca na projekte – TODO

V rámci práce na projekte sme objavili v kód objavili viaceré komentáre typu TODO, teda komentáre, ktoré obsahovali informácie o práci, ktorá by v rámci danej časti projektu mala byť ešte dokončená alebo vypracovaná. Preto sme sa rozhodli tieto TODO komentáre v kóde vyhľadať, analyzovať ich význam a následne ich vypracovať. Avšak mnohé TODO komentáre mali buď nedostatočný popis, týkali sa častí projektu, ktoré ešte nie sú používané, alebo už boli vypracované avšak nikto neodstránil komentár TODO. Preto sme TODO rozdelili do viacerých skupín

4.3.1. Vypracované úlohy TODO

Tieto úlohy TODO boli riadne vypracované a TODO komentáre boli odstránené. Jedná sa o nasledovné úlohy:

TODO Calculate if two lines are part of central circle

Je potrebné zapracovať metódu či sú dve čiary súčasťou stredového kruhu do triedy *JustLineSeen*. Taktiež je potrebná úprava komentárov pre jednotlivé, už existujúce metódy v danej triede.

TODO Communication test not finished. Insert testing parameters

V projekte sú len 2 miesta, kde sa daná trieda vyskytuje. Z dostupnej dokumentácie nie je jasné, ako tím pôvodne myslel implementáciu danej triedy. Po dohode s vedúcim sme danú triedu zmazali.

TODO effectors into HashMap

Effectory sú uchovávané v ArrayListe, no dopytujú sa podľa Stringu, preto by bolo vhodné zvoliť inú dátovú štruktúru, takže sme zvolili HashMapu kvôli rýchlosti dopytu hodnôt.

TODO Premyslieť, ako sa bude riešiť zlá nadväznosť pohybov

Nadväznosť pohybov je riešená v triede sk.fiit.jim.agent.trajectory.TrajectoryPlanner.java a bolo ju treba upraviť. Podmienka na znovuplánovanie trajektórie po prejdení ¾ vzdialenosti bola odstránená a vola navrhnutá zmena na vykonávanie trajektórie.

TODO Implement singleton in SkillsFromXmlLoader.java

Funkcionalita triedy SkillsFromXmlLoader.java je používaná na mnohých miestach v projektu, vždy však pracujeme s novou inštanciou. Návrhový vzor singleton zabezpečuje, že vždy pracuje s nanajvýš jednou inštanciou triedy.

TODO opravit pripad, ked je size 0

Bola identifikovaná exception v triede AgentPositionCalculator keď funkcia dostala veľkosť 0 a bola opravená.

TODO refactor into class hierarchy

V triede Preceptors.java bol zbytočne dlhý if, ktorý bol nahradený vhodnejším a rýchlejším riešením cez switch.

TODO better check two lists ? Maybe HASH ?

V kóde sa nachádzalo prehľadávanie dvoch listov foreach cyklami, čo sa javilo ako neefektívne riešenie. Bolo implementované nové riešenie pomocou údajovej štruktúry Set.

TODO implement the prediction module correctly

Ide to vypracovanie jedného z TODO, ktoré boli pridané niektorým z predošlých tímov. Úloha sa viaže k triede sk.fiit.jim.agent.models.prediction.Prophet, V triede neboli vykonané žiadne zmeny ale zatiaľ ju nevymažeme, pretože by mohla byť použitá v budúcnosti.

TODO change conditions in method sequenceCheck

V triede TrajectoryPlanner a MoveValidator sme zistili, že metóda sequenceCheck nie je úplná a mala v sebe priestor na zmeny – úpravu podmienok pri zlej nadväznosti pohybov.

TODO treba zapocitat odchylku

V triede GameView bolo potrebné na viacerých miestach pridať do výpočtov odchylku, keďže hodnoty zo server nie sú úplne presné. Na riešenie odchylky bola vytvorená konštanta TOLERANCE, ktorej hodnota bola po testovaní nastavená na 0.1, pričom táto premenná slúži ako odchylka.

TODO move to RoboCupLibrary

JV Triede sk.fiit.jim.agent.AgentInfo sa nachádza metóda *isInHalfPlane*(), ktorá by mala byť presunutá do modulu RoboCulLibrary. Metóda vypočitáva, či bod typu *Vector3D* spadá do polroviny. Metóda bola presunutá do triedy sk.fiit.robocup.library.geometry.Vector3D a bola upravená, aby používala súradnice z objektu, nie zo vstupného vektora..

TODO referencie na iné kontrolery

Jedná sa o vypracovanie idenfitikovaného TODO. V Triede *TabTournamentController* sa nachádza metóda *updateTournamentTab*(), ktorá by mala byť presunutá do triedy *sk.fiit.testframework.ui.controllers.MainFrameController*. Metóda bola presunutá do triedy *sk.fiit.testframework.ui.controllers.MainFrameController*.

TODO try to create solution with reusable Singleton Factory

Jedná sa o implementovanie snávrhového vzoru singleton v rámci triedy MovementSkills. Pri testovaní však bolo odhalené, že pri konkrétnom použití na podtriedy typu Walk hráč nefunguje správne.

4.3.2. Nevypracované úlohy TODO

Tieto úlohy TODO neboli z časových dôvodov vypracované a ostávajú ako práca na projekte do budúcnosti.

TODO implement calculation of opponent position

Class: sk.fiit.jim.agent.models.DynamicObject.java

- Popis ku konkrétnemu TODO sa nenachádzal v žiadnej z dokumentácii, no tím SixPack sa vo svojom inžinierskom diele zaoberal touto problematikou na 30. Strane

TODO a deviation has to be considered

Class: sk.fiit.jim.agent.models.TacticalInfo.java

- Výpočet sa vykonáva len v dvojrozmernom priestore, je potrebné výpočty aktualizovať do trojrozmeného priestoru.

TODO accelerometer is relative to torsoPosition, not centerOfMass

Class: sk.fiit.jim.agent.models.AgentModel.java

TODO calculating who is on our team and who is the opponent.

Class: sk.fiit.jim.agent.models.WorldModel.java

TODO does it work? should it work this way?

Class: sk.fiit.jim.agent.skills.HighSkill.java

- Komentár je and metódou checkProgress, ktorou jedinou úlohou je vyhodiť výnimku

TODO is there some better solution?

Class: sk.fiit.testframework.AnnotatorTestCase.java

- V metóde destroy v nútri try-catchu, jediné čo sa deje je uspatie vlákna na 2 sekundy.

4.3.3. Už vypracované TODO úlohy

V tejto časti sa nachádzajú TODO, ktoré už boli vypracované, avšak komentáre TODO neboli odstránené .

TODO consider relativeDistanceToTarget, consider annotation's min_distance

- TODO už bolo vypracované a nie je potrebná nová implementácia.

TODO logic for scoring of result

Class: sk.fiit.testframework.moveoptimizator.MoveOptimizer.java

- TODO už bolo vypracované.

TODO: Replace with purpose of method. Start with verb

Class: sk.fiit.jim.annotation.data.MoveValidator.java

- TODO už bolo vypracované.

4.3.4. TODO odstránené kvôli nejasnosti alebo nepotrebnosti

Tieto TODO boli buď vyhodnotené ako nepotrebné pre projekt, alebo boli príliš nejasné a chýbal podrobnejší popis umožnujúci vypracovať dané TODO.

TODO send an exit command to agents not launched by the test framework

Class: *sk.fiit.testframework.communication.agent.AgentManager.java*:

- Na základe konzultácií na stretnutí s vedúcim projektu sme sa dohodli o zamietnutí/nevypracovaní tejto úlohy, pretože dôležitosť riešenia tejto úlohy nie je podstatná a nemá dopad resp. prínos pre iné úlohy, ktoré riešime.

TODO Only localhost now | is that still true???

Class: sk.fiit.testframework.communication.agent.AgentJim.java

- Po konzultácií s vedúcim bolo vyhodnotené TODO ako nepotrebné, keďže v prípade, ak agent nebeží na localhoste, tak nestíha spracovávať správy od serveru v požadovanom čase

TODO pokracovat

Class: sk.fiit.testframework.ui.controllers.TabAgentTrainingController.java

- K TODO úlohe chýbal akýkoľvek popis, takže TODO bolo odstránené na základe nejasnosti úlohy

TODO GET CHECKSUM FROM PLAYER

Class: sk.fiit.testframework.annotator.Annotator.java

- K TODO úlohe chýbal akýkoľvek popis, takže TODO bolo odstránené na základe nejasnosti úlohy

TODO: is this safe (no)? is it called from the test framework?

Class: sk.fiit.jim.agent.skills.HighSkill.java

- TODO bolo vyhodnotené ako zbytočné na vypracovanie

TODO moves/walk_dynamic.xml

Class: sk.fiit.jim.agent.highskill.move.WalkFastZMPOld.java

- K TODO úlohe chýbal akýkoľvek popis, takže TODO bolo odstránené na základe nejasnosti úlohy
- Taktiež sa TODO nachádza v "old" triede, takže ďalšie skúmanie zmyslu TODO bolo vyhodnotené ako nepotrebné

4.3.5. TODO vypracované upravením komentáru

Tieto úlohy TODO boli analyzované, avšak v momentálnom stave projektu by implementácia TODO nemohla byť korektne otestovaná, preto boli vypracované iba bližsou analýzou a upravením komentáru TODO v kóde.

TODO compute relative fitness to strength of kick (annotations)

Hodnota Fitness v metóde getFitnessPass v triede TacticalInfo slúži na výpočet fitness funckie vyjadrujúcej pravdepodobnosť úspešnosti prihrávky. Hodnota je počítaná len z uhlu medzi smerom prihrávky a protihráčmi, ale do hodnoty by mala byť započítaná aj sila kopu.

TODO nova verzia ma v nastaveniach Agent Radius 0.4. Je otazne z coho je tato hodnota 0.2 urcena.

Premenná ROBOT_RADIUS v triede Obstacles je inicializovaná a používaná s hodnotou 0.2. Bolo by potrebné otestovať ako robot vyhodnocuje prekážky pre hodnoty 0.2 a 0.4 a na základe testovania upraviť hodnotu premennej.

Class: sk.fiit.jim.agent.AgentInfo.java

TODO Agent's goal is allways left, also during second period

Jedná sa o premennú side v triede AgentInfo. Hráč útočí na ľavú bránku počas oboch polčasov. Premenná side závisí od iných premenných, konkrétne OUR_SIDE_IS_LEFT a HALF_TIME_CHANGE_SIDES.

4.4. Chýbajúca dokumentácia

4.4.1. TestFramework

Počas práce na projekte, sme konvertovali projektu do Maven profilu, čo však spôsobilo znefunkčnenie pridávanie agenta v Testframeworku. Identifikovali sme, že to bolo spôsobené nesprávnymi parametrami v príkaze spúšťania.

Parametre boli opravené a logika pridávania agenta do TF sme lepšie opísali na Wiki.

4.4.2. Matematický model pohybu a polohy

Zero Moment Point

ZMP sme v kóde identifikovali na troch miestach: agentovi – highskill – move. Rovnako sme na začiatku tímového projektu nachádzali v kóde množstvo pripomienok „To DO“, ktoré sme sa snažili počas práce na projekte vyriešiť, avšak častokrát, chýbal detailnejší popis, čo touto poznámkou mal za cieľ predošlí tím vyriešiť.

Trieda StabilityWalkTactic bola okomentovaná.

Trieda WalkFastZMP zapoznámkovaná nebola vôbec okomentovaná.

Triedy v zložke *move* sú veľmi slabo zdokumentované a ťažko sa v nich orientuje.

Tieto informácie sme uložili aj na wiki, pre budúci tímový projekt.

Kalmanov filter

Kalman filter bol implementovaný predošlým tímom BAREKO. Je implementovaný v triede KalmanReal, v balíku sk.fiit.jim.agent.kalman.

Náš tím analyzoval potencionálnu optimalizáciu Kalman filtra, avšak predošlí tím neuvádza žiadnu informácie o možných priestoroch optimalizácie ani nijak hlbšie vedenú dokumentáciu k možným zmenám. Informácie, ktoré sme ku Kalman filtru analyzovali a zistili, sme uložili na wiki, pre budúci tímový projekt. Žiaľ, počas projektu sme neprišli do styku s hlbšími informáciami či dokumentáciou z predošlých tímov, preto sme na wiki obsiahli len naše získané vedomosti.

Výmena súradníc X,Y

Tím WAWOT analyzoval aj chybnú implementáciu súradní X a Y. Minuloročné tímy, neevidovali podrobnejšiu dokumentáciu pri tejto implementácii, ako aj dokumentáciu samotného kódu v podobe základných komentárov.

Pri analyzovaní tohto problému sme identifikovali chýbajúcu dokumentáciu, ktorá by nám pomohla hlbšie pochopiť princíp implementácie a následne umožnila opraviť výmenu súradníc.
- wiki a články ohľadom výpočtu sférických a karteziánskych súradníc sú neaktuálne, staré a neúplné

LowSkills

V metóde *NewActivePhase()* triedy *LowSkill* sa nachádza komentár: *//TODO possible stack overflow if all phases have the same skipIfFlag* Ku ktorému neevidujeme žiadnu bližšiu informáciu, za akým účelom ju predošlí tím pridal do kódu.

5. Spracovanie oblastí projektu

5.1. Analýzy, návrhy a implementácie

5.1.1. Analýza java kódu (yourkit)

Predošlý tím identifikoval ako jeden z problémov neefektívnosť momentálneho riadiaceho kódu, ktorý sa z určitej časti podieľa na malej výkonnosti hráča. Ide hlavne o časti, ktoré sa starajú o spracovávanie komunikácie so serverom a následného ovládania hráča. Cieľom analýzy je zistiť konkrétne problémy v neefektívnosti kódu a navrhnúť vylepšenia, ktoré budú viesť k jeho optimalizácii.

5.1.1.1. Nástroj Yourkit Java Profiler

Tento nástroj pomáha pri sledovaní výkonnosti a debugovaní Java aplikácií na základe monitorovania operácií na JVM úrovni. Pomocou rôznych techník sleduje čas procesora pri jednotlivých metódach a vláknach, tvorbu objektov, alokáciu pamäte apod. Yourkit ponúka hlavne prepracovaný CPU profiling, ktorý môžeme využiť na analýzu náročnosti metód a ich optimalizácie.

Profiling ponúka 3 možnosti:

• Sampling

• Tracing

• Call Counting

Pre najpresnejšie zmeranie spotrebovaného času volanými metódami je najvhodnejšia Tracing metóda, čo vyplýva z následujúceho obrázka.

Stop CPU Profiling

	Description	Overhead	Time accuracy	Invocation counts	Call stacks
Sampling	Estimate method times by periodically probing stacks of running threads CPU sampling settings...	Low	High for long, low for short methods	No	Yes
Tracing	Accurately measure method (*) times and invocation counts by monitoring enters and exits CPU tracing settings...	Low to high (**)	Usually high (**)	Yes	Yes
Call counting	Count method invocations (*)	Very low	Time is not measured	Yes	No

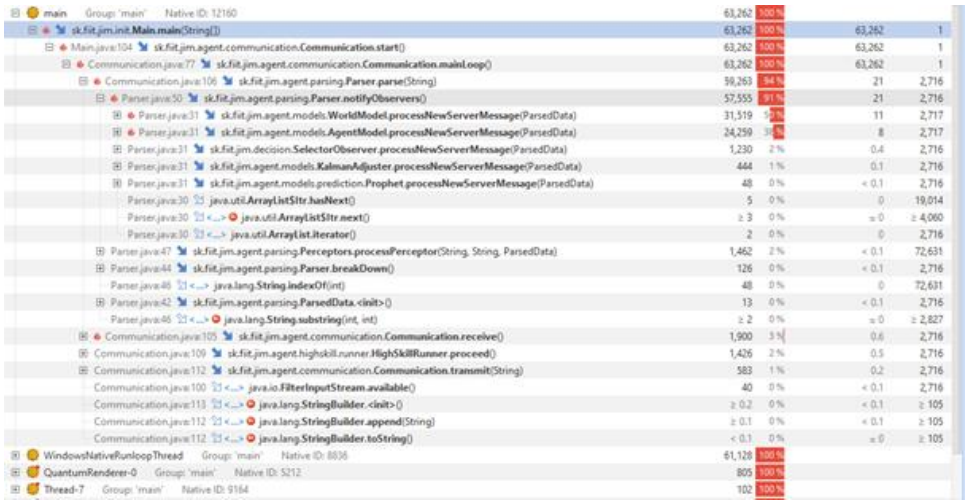
(*) - CPU tracing and call counting skip trivial methods by default - see Help for detail
(**) - depends on applied CPU tracing settings and profiled application

Obrázok 1 Výstup analýzy Yourkitom

Yourkit takisto po dostatočne dlhom behu programu ponúka uloženie zozbieraných výsledkov vo forme snapshotov, ktoré budú pre analýzu vhodnejšie, než sledovanie výsledkov aktuálneho behu.

5.1.1.2. CPU profiling hráča

Pomocou spomínanej metódy Tracing sme zmerali celkový čas jednotlivých vykonaných metód. Výstupom sú teda zoradené metódy podľa času s ohľadom na počet ich invokácií.



Obrázok 2 CPU Profilling hráča

Pomocou tejto štatistiky môžeme ďalej analyzovať vykonávanie programu a hlbším vnorením do metód zistiť, aké kúsky kódu sú pre CPU časovo najnáročnejšie.

5.1.1.3. Vykonané metódy počas behu hráča

Ako je z obrázku zrejmé, spracovanie komunikácie, aktualizovanie stavu hráča a príkazov prebieha iba pomocou jedného vlákna v dôsledku čoho pravdepodobne dochádza k zahadzovaniu prijatých správ, ktoré agent prijme počas spracovávania inej správy. Toto spracovanie trvá väčšinu spotrebovaného času, čomu sa však nedá vyhnúť. Veľkým zlepšením by ale mohlo byť vytvorenie nového vlákna pre spracovanie každej novej správy. V rámci ostatných metód bude optimalizácia spočívať v nahradení jednotlivých kódových konštrukcií výpočtovo menej náročnými kúskami kódu.

5.1.1.3.1. Metóda Communication.mainLoop()

Táto metóda beží od začiatku pripojenia k serveru a spracováva prijaté správy o stave hráča z pohľadu servera a teda zaberá drvivú väčšinu celkového času.

```
private void mainLoop() throws IOException {
   while(true){
      // wait for input
      while (input.available() == -1) {
         Thread.yield();
      }

      // receive message
      String incoming = receive();
      parser.parse(incoming);

      // process message
      HighSkillRunner.proceed();

      // transmit message
      transmit(outMessageBuffer.append("(syn)").toString());
      outMessageBuffer = new StringBuilder();
   }
}
```

Obrázok 3 Zdrojový kód metódy Communication.mainLoop

V tejto metóde je ďalej volané spracovanie správy pomocou metódy *Parser.parse(incoming)*. Vytvorenie nového vlákna pre každé toto volanie by mohlo priniesť zlepšenie v celkovej rýchlosti spracovávania správ a tým pádom bude mať hráč presnejšie informácie o jeho stave.

5.1.1.3.2. Metóda Parser.parse(String)

V rámci metódy dochádza k rozloženiu správy na jednotlivé informácie, ktoré server ponúka. Ďalej sa tieto informácie preložia do formy dát, ktoré sa ďalej dajú v programe spracovávať. Nakoniec sa zavolá aktualizovanie stavu objektov, ktoré ovládajú hráča.

```
public ParsedData parse(String message) {
    data = new ParsedData();
    this.message = message;
    String[] breakDown = breakDown();
    for (String perceptor : breakDown) {
        String perceptorId = perceptor.substring(0, perceptor.indexOf(' '));
        Perceptors.processPerceptor(perceptorId, perceptor, data);
    }

    notifyObservers();
    return data;
}
```

Obrázok 4 Zdrojový kód metódy Parser.parse

Priestoru na optimalizáciu veľa nie je, keďže obsiahnuté metódy až na *Parser.notifyObservers()* podľa Yourkitu príliš náročné nie sú.

5.1.1.3.3. Metóda Parser.notifyObservers()

V tejto metóde sa už volajú konkrétne objekty, ktoré sa starajú o spracovávanie preložených dát a aktualizovanie stavu hráča alebo hracieho sveta a ovládajú jeho správanie.

```
private void notifyObservers() {
    synchronized (Parser.class){

        for (ParsedDataObserver observer : observers)
            observer.processNewServerMessage(data);
    }
}
```

Obrázok 5 Zdrojový kód metódy Parser.notifyObservers

Priamo v metóde nie je možnosť nejakého optimalizovania, to bude možné až pri spracovávaní správ objektami *observer*.

5.1.1.3.4. Metóda WorldModel.processNewServerMessage(ParsedData)

Táto metóda je jedna z dvoch, ktoré sú podľa Yourkitu medzi spracovávaním objektami *observer* časovo najnáročnejšie.

```
public void processNewServerMessage(ParsedData data) {
    if (data.ballRelativePosition != null) {
        calculateBallPosition(data);
    }

    calculatePlayers(data.otherplayers, data.SIMULATION_TIME);

    if (data.lines.size() > 0) {
        calculateLines(data);
    }

    try {
        TestFrameworkCommunication.sendMessage(new
Message().WorldModel().changed(WorldModel.getInstance()));
    } catch(Exception e) {
        LOG.log(LogType.WARNING, "Chyba v spracovani spravy vo WorldModeli",
e);
    }
}
```

Obrázok 6 Zdrojový kód metódy WorldModel.processNewServerMessage

V metóde sa potom vypočítava aktuálny stav hracích tímov, ich pozície a pozícia lopty. Tieto metódy však nie sú kritické. Ďalej sa tu ešte vykonáva informovanie testovacieho frameworku o aktuálnom stave, čo je z pohľadu času kritické. Vypnutím tohto odosielania keď nie je potrebné by sme mohli ušetriť nejaký čas.

5.1.1.3.5. Metóda Message.WorldModel.Changed(WorldModel)

Pomocou tejto metódy sa vytvorí správa o aktuálnom stave hracieho sveta, ktorá sa odošle do testovacieho frameworku.

```
public Message changed(sk.fiit.jim.agent.models.WorldModel model) {
    message += " beginData\n";

    try {

        ByteArrayOutputStream outStream = new ByteArrayOutputStream();
        ObjectOutputStream out = new ObjectOutputStream(outStream);
        out.writeObject(model);
        out.close();
        byte[] data = outStream.toByteArray();
        String to_send = Base64.encodeBase64String(data);
        outStream.close();
        MessageDigest md = MessageDigest.getInstance("SHA-1");
         Formatter formatter = new Formatter();
         for (byte b : md.digest(data)) {
             formatter.format("%02x", b);
         }
         String checksum = formatter.toString();
         message += checksum + "\n";
        message += to_send;
        message += "endData";
        formatter.close();

    } catch (Exception e) {
        LOG.log(Level.WARNING, "Couldnt process the message about the world.
Cause: " + e.getMessage());
    }

    return Message.this;
}
```

Obrázok 7 Zdrojový kód metódy Message.WorldModel.Changed

V metóde je najkritickejší zápis dátového objektu do bajtového output streamu. Potom je tu náročné ešte kódovanie Base64 stringu a transformácia na jeho hash hodnotu s SHA-1. Tieto samotné metódy sa však príliš zmeniť nedajú.

5.1.1.3.6. Metóda AgentModel.processNewServerMessage(ParsedData)

V tejto metóde sa prepočítava stav tela hráča, pozícia častí a natočenie. Po jej zbehnutí by mal hráč mať informácie o svojej pozícii a orientácii.

```
public void processNewServerMessage(ParsedData data) {
    if (data.PLAYER_ID != null && data.PLAYER_ID != 0) {
        AgentInfo.getInstance().setPlayerId(data.PLAYER_ID);
    }

    if (data.OUR_SIDE_IS_LEFT != null) {
        AgentInfo.getInstance().setAssignedSide(true);
        AgentInfo.side = data.OUR_SIDE_IS_LEFT ? Side.LEFT : Side.RIGHT;
    }

    /*
     * According to values and server settings, if sides change at half
     * time, side is changed. Roman Moravcik (Gitmen)
     */
    if (HALF_TIME_CHANGE_SIDES && !sidesChanged && data.GAME_TIME >=
HALF_TIME) {
        sidesChanged = true;
        AgentInfo.side = AgentInfo.side == Side.RIGHT ? Side.LEFT :
Side.RIGHT;
        LOG.log(LogType.AGENT_MODEL, "Sides changed at half time");
    }

    deleteHistory(data);

    lastDataReceived = data;
    lastAccelerometer = data.accelerometer;

    updateJointPositions(data);
    adjustRotationsFor(data.gyroscope);
```

```
    updateRotations(data);
    updatePureBodyAcceleration(data);

    if(extendHistory(data))
        updatePosition(data);

    // Following methods are used for Zero moment point
    updateBodyPartsPositions2();
    updateCenterOfMass();
    updateFeetForce(data);
    updateZeroMomentPoint();
    updateSpeedX(data);
    if (data.fixedObjects != null && data.fixedObjects.size() > 0)
        lastTimeFlagSeen = data.SIMULATION_TIME;
    // End zero moment point

    updateHistory(data);
}
```

Obrázok 8 Zdrojový kód metódy AgentModel.processNewServerMessage

Časovo náročná je tu metóda *AgentModel.updateBodyPartsPositions2()*, v ktorej sa nachádzajú vektorové výpočty avšak najviac času tu zaberá výpis do logu. Odstránením tohto výpisu by sme mali ušetriť nejaký čas.

Pomocou nástroja Yourkit sme mohli identifikovať metódy, ktoré sú počas behu programu časovo najnáročnejšie a bližšie pochopiť, ktoré konkrétne časti sú za to zodpovedné. Na základe týchto poznatkov dokážeme niektoré kódové konštrukcie optimalizovať a zlepšiť tým celkový výkon hráča. V prvom rade ide o zavedenie viacerých vlákien pri spracovávaní komunikácie so serverom a odstránenie niektorých funkcií, ktoré nie sú pre beh programu nevyhnutné. Všetky tieto zmeny sa budeme snažiť v kóde implementovať a tým by sa mal problém s výkonnosťou do určitej úrovne zmierniť.

5.1.2. Analýza súčasného riešenia orientácie Jima

Jim predstavuje Java aplikáciu samotného agenta, ktorý sa pripojí k serveru a hrá futbal. Agent by mal dodržiavať nakonfigurovanú a naprogramovanú taktiku. Pred spustením agenta je nutné spustiť RCSS server. Aplikácia agenta obsahuje jednoduché ovládanie prostredníctvom GUI umožňujúce opätovné načítanie pohybov a preplánovanie. Inicializačným bodom agenta je trieda sk.fiit.jim.init.Main.

Agent dokáže kopať do lopty, otočiť sa o 60 stupňov, postaviť sa za maximálne za 3s, stabilizovať sa pri vstávaní, behať, chodiť. Taktiež má agent naimplementované základné vnímacie taktiky (či tím útočí alebo bráni).

Spustenie agenta je pomerne zdĺhavý proces. Na počítači so slabším výkonom čas spustenia sa pohybuje cca. 10 sekúnd. Na začiatku spúšťania Jima sa vždy vykonajú nasledovné operácie:

- Načítavanie pohybov z XML súborov v adresári moves
- Načítanie anotácií
- Spracovanie parametrov príkazového riadku
- Spustenie TFTP servera použitého pre komunikáciu s TestFrameworkom
- Prípadné vytvorenie GUI
- Vstup do hlavného cyklu

Správanie agenta je riadené kódom, a možno ho rozdeliť do niekoľkých vrstiev.

5.1.2.1. Pohyb

Prvou vrchnou vrstvou je plánovač riadený triedou HighSkillPlanner nachádzajúci sa v balíku sk.fiit.jim.agent.highskill.runner. Plánovač vytvára inštancie tzv. high skillov (vyšších pohybov). High skilly počas svojho behu vyberajú nižšie pohyby (nižšia vrstva) tzv. low skill, ktorý sa má vykonať. Low skill predstavuje len skupinu metadát, zapísaných pomocou XML, pričom určujú jeho názov a ďalšie informácie.

SkillsFromXmlLoaderhe trieda, slúži na načítanie pohybov z XML súborov.

Triedy LowSkills a Phases spravujú objekty fáz a pohybov a sú umiestnené v balíku sk.fiit.jim.agent.moves.

Keďže načítanie pohybov je značne pomalé, ukladá sa ich parsovaná podoba do súboru ./movecache, ktorý sa pri budúcom načítaní použije, pokiaľ od jeho vytvorenia nebol žiadny pohyb zmenený. V takom prípade sa súbor vytvorí nanovo.

Predchádzajúci tím obohatil knižnicu RoboCupLibrary efektívnejšími výpočtami sin a cos funkcií, ktoré sú implementované v celom projekte. Triedy z knižnice sa využívajú ako v Test frameworku tak aj v agentovi. Cieľom je čo najviac odstrániť tieto závislosti medzi Jim-om a TestFrameworkom, čo však v súčasnosti nie je dodržané.

V agentovi sa na komunikáciu s TestFrameworkom využíva trieda TestFrameworkCommunication, ktorá sa nachádza v balíku sk.fiit.jim.agent.communication.testframework.

5.1.2.2. Hlava

Otáčanie hlavy funguje na princípe xml súborov, ktoré sa nachádzajú v priečinku move v Jimovi. Xml súbory obsahujú radu parametrov, pre otáčanie jednotlivých kĺbov Jima. Pohyby hlavy sú reprezentované súbormi:

- head_left_120.xml
- head_right_120.xml
- head_left_down.xml
- head_right_down.xml

Súbory head_left_120.xml a head_right_120.xml zabezpečujú otáčanie hlavy do strán pod maximálnym uhlom 120° a súbory head_left_down.xml a head_right_down.xml zabezpečujú náklony hlavy nadol a nahor.

Tím Bareko uvádza, že analýzou predchádzajúcich tímov sa priklonili k uváženiu vylúčiť pohyb hlavy nahor a nadol z dôvodu, nakoľko tento pohyb signalizoval nulovú pridanú hodnotu pri vnímaní prostredia či lopty. Tím taktiež analyzoval, že Jimova hlava pri chôdzi z časti mení svoju rotáciu vzhľadom na celé telo, no táto rotácia v priestore sa pri vnímaní čiar nijako nezohľadňuje.

Tím Bareko venoval pozornosť aj dekrementovaní vzdialenosti hráča od lopty pri zmene rýchlosti chôdze, čo sa im podarilo aj úspešne implementovať. Táto implementácia však nemala žiaden vplyv na pády Jima počas chôdze, plus tejto implementácie je zníženie času na presun k lopte.

Na určenie vzdialenosti hráča od lopty, pri ktorej hráč spomalí svoj pohyb k lopte sa používa konštanta EDGE_DISTANCE_FROM_BALL v balíku sk.fiit.jim.decision.situation, konkrétne v triede Situation. Hodnota tejto premennej je 1.5.

5.1.3. Analýza zisťovania polohy a orientácie hráča

Správa o polohe Jima sa skladá z nasledujúcich údajov: poloh lopty, poloha hráča, vypočítanie čiara a následné odoslanie správy do test frameworku.

Orientácia Jima sa odohráva v triede VECTOR3D. V triede AgentPositionCalculator zisťuje, kde sa agent nachádza.

Na základe rozpoznávania vzorov čiar, Jim vie definovať na čo sa pozerá podľa preddefinovaných vzorov. Jim rozoznáva nasledovné vzory v triede LinePatternRecognition, ktorá poskytuje funkcie na klasifikáciu vzťahov medzi čiarami a bodmi.:

- **V triede sú naimplementované nasledujúce funkcie:**

squareDistance (druhá mocnina vzdialenosti medzi bodmi)

```
private static double squareDistance(Vector3D a, Vector3D b){
    return (a.getX()-b.getX())*(a.getX()-b.getX()) +(a.getY()-b.getY())*(a.getY()-b.getY()) +
}
```

```
+ (a.getZ()-b.getZ())*(a.getZ()-b.getZ());
```

Obrázok 9 Zdrojový kód metódy squareDistance

IsSamePoint

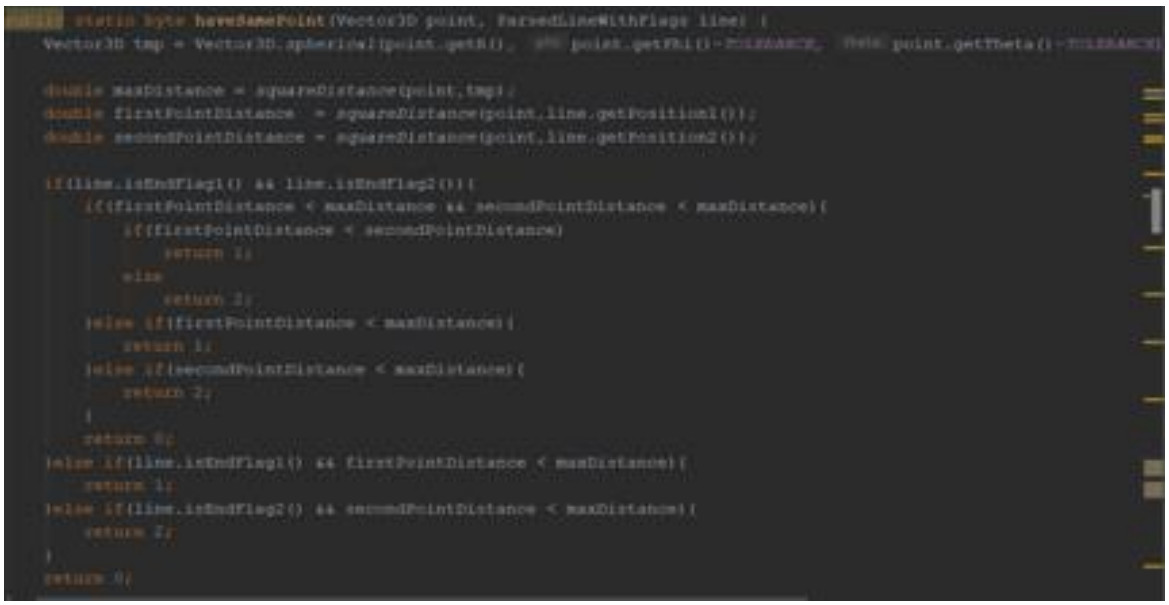
funkcia najskôr vypočíta maximálnu vzdialenosť medzi dvoma bodmi s najhorším možným prípadom (prvý point noise pre oba uhly je +2 a pre druhý je -2. Ak je vzdialenosť medzi dvoma danými bodmi menšia ako najhorší možný prípad, oba body by mohli predstavovať rovnaký bod. Návratová hodnota @return je true, ak oba body môžu byť rovnaké - agent vidí jeden bod s dvoma rôznymi úrovňami).

```
public static boolean isSamePoint(Vector3D a, Vector3D b) {
    Vector3D tmp = Vector3D.spherical(a.getR(),  phi: a.getPhi()-TOLERANCE,  theta: a.getTheta()-TOLERANCE);
    double distance = squareDistance(a,tmp);
    double pointDistance = squareDistance(a,b);
    return (distance > pointDistance);
}
```

Obrázok 10 Zdrojový kód metódy isSomePoint

haveSamePoint (vzor vlajka a čiara)

Táto funkcia používa ako vstupné parametre Vector3D point a ParsedLineWithFlags line. Úlohou tejto funkcie je zistiť, ktorý z bodov čiary môže byť totožný s bodom predaným ako argument. Funkcia má tri návratové hodnoty 0(ak žiaden bod na čiare nie je totožný), 1 (ak prvý koncový bod riadku a daný bod by mohol predstavovať ten istý bod) a 2(ak druhý koncový bod riadku a daný bod by mohol predstavovať rovnaký bod).



Obrázok 11 Zdrojový kód metódy haveSamePoint

isCorner (vzor roh) Ako vstupné parametre sú používané ParsedLineWithFlags line1 ParsedLineWithFlags line2. Funkcia zisťuje či tieto parametre predstavujú roh na ihrisku alebo nie.

Obrázok 12 Zdrojový kód metódy isCorner

Ak je uhol menší ako 117° (v polovici medzi 90 a 144), tak sa jedná sa o roh, inak ak je väčší ako 117° jedná sa o časť kruhu.

Z implementácie môžeme vidieť, že funkcia využíva samePoint funkciu, pre určenie návratovej hodonty isCorner funkcie.

areCircleLines(časť kruhu) Implementácia tejto funkcie je zhodná s getCorner, nakoľko sa jedná len o rozdiel v definovaní kedy sa jedná o časť kruhu. Ak dve čiary majú spoločný koncový bod a zvierajú uhol 144 stupňov, vieme definovať, že sa jedná o kruh, ak je tento uhol väčší ako 117 stupňov, jedná sa o časť kruhu. Tolerancia je 27stupňov.

goalAndLine (či daná čiara je v blízkosti bránky - vypočítava sa na základe bránkovej tyče) Vstupné parametre sú Vector3D point ParsedLineWithFlags line. Návratová hodnota je vzdialenosť medzi ortogonálnym projekciou bodu a koncovými bodmi čiary.

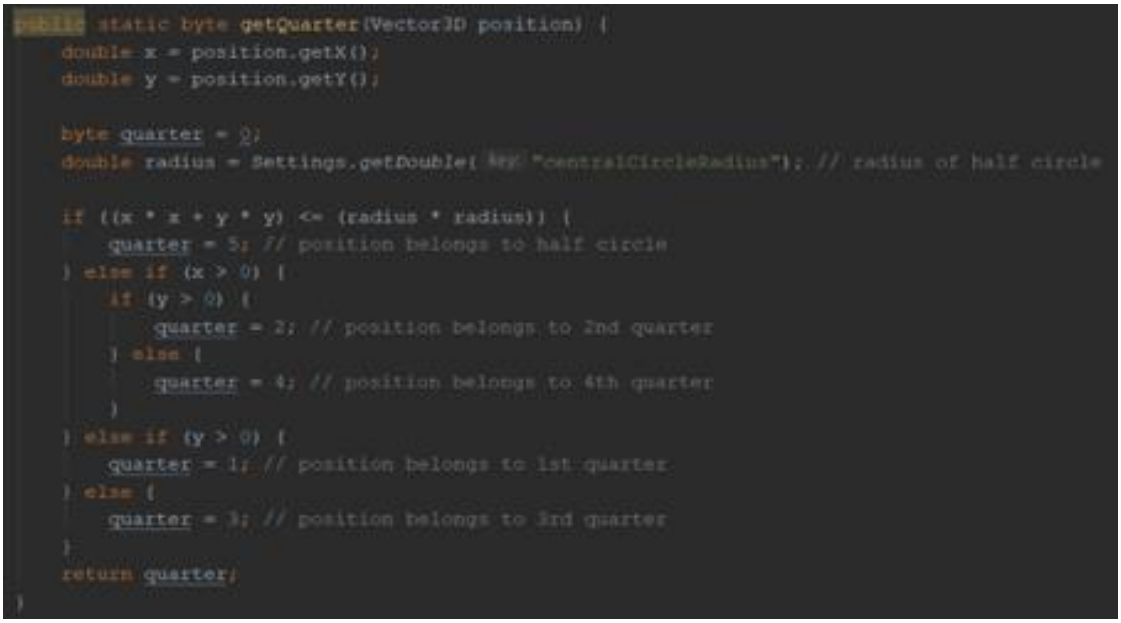
vzdialenosti [0] = vzdialenosť medzi prvým bodom trate a ortogonálnym premietnutím bodu na čiare.

vzdialenosti [+] = vzdialenosť medzi druhým bodom trate a ortogonálnym premietnutím bodu na čiare.

Obrázok 13 Zdrojový kód metódy goalAndLine

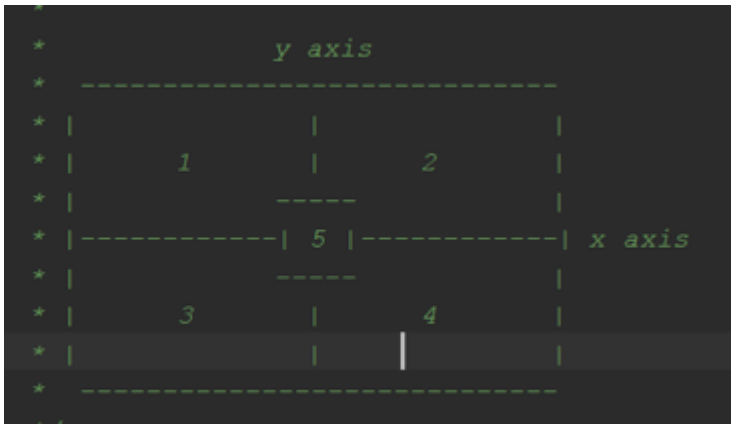
Metóda getQuarter:

Sa nachádza v public class *AgentPositionCalculator*, ktorá vypočíta aproximáciu aktuálnej pozície agenta.



Obrázok 14 Zdrojový kód metódy getQuarter

V metóde getQuarter definujeme rozdelenie ihriska na 5 základných sektorov. Tieto sektory sú nápomocné Jimovi pri jeho orientáci, nakoľko na základe hodnoty súradníc (x,y - záporné alebo kladné) Jim vie rozpoznať v ktorom z piatich základných sektorov sa nachádza.



Obrázok 15 Metóda Rozdelenie ihriska na 5 základných sektorov

5.1.3.1. Návrhy na implementovanie

isIntersection (funkcia na zistenie či čiary majú spoločný priesečnik: T alebo +)

isT (čiary s priesečníkom T)

isPlus (čiary s priesečníkom +)

isGoalBox (či sú body bránkovisko)

Na základe aktuálneho stavu implementácie, s ktorou začíname pracovať v rámci nášho projektu, vieme zhodnotiť, že Jim sa vie orientovať, vie zistiť v ktorej pätine ihriska sa nachádza, avšak niektoré špecifické orientačné body sú len vo fáze dokumentácie a bude potrebné ich doimplementovať, ako napríklad zistenie či sa jedná o bránkovisko alebo stredovú časť ihriska.

5.1.4. Analýza matíc u iných hráčov

Najlepší tím sveta – UA Austin Villa má riešenie svojej pozície a transformácií zverejnené. Kód na menenie pozícií ich vektorov sa nachádza v súboroch *utaustinvilla.cc* a *utaustinvilla.h*. Ďalej používajú rôzne geometrické výpočty v súboroch: *utaustinvillageom.cc, utaustinvillageom.h, utaustinvillamat.cc* a *utaustinvillamat.h*.

Vektory na otáčanie majú riešené nasledovne:

```
VecPosition VecPosition::rotateAboutX(double angle) {

    double cos = cosDeg(angle);
    double sin = sinDeg(angle);

    double newX, newY, newZ;
```

```
    newX = m_x;
    newY = m_y * cos + m_z * sin;
    newZ = -m_y * sin + m_z * cos;

    return VecPosition(newX, newY, newZ);
}

VecPosition VecPosition::rotateAboutY(double angle) {

    double cos = cosDeg(angle);
    double sin = sinDeg(angle);

    double newX, newY, newZ;

    newX = m_x * cos - m_z * sin;
    newY = m_y;
    newZ = m_x * sin + m_z * cos;

    return VecPosition(newX, newY, newZ);
}

VecPosition VecPosition::rotateAboutZ(double angle) {

    double cos = cosDeg(angle);
    double sin = sinDeg(angle);

    double newX, newY, newZ;

    newX = m_x * cos + m_y * sin;
    newY = -m_x * sin + m_y * cos;
    newZ = m_z;

    return VecPosition(newX, newY, newZ);
}
```

Tím FC Portugal má zverejnené pár publikácií, ako napríklad *Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents*, kde je opísané, ako riešia svoje pozície. Našli sme ich jeden zdrojový súbor hráča z roku 2000, kde sa nachádzajú nasledovná implementácia menenia pozície:

```
ActionQueueRes go_to_point(Vector p, float buffer, float dash_power, DodgeType
dodge) {
    Mem - > LogAction5(30, "go_to_point %d (%.1f %.1f)", dodge, p.x, p.y);
    if (!Mem - > MyConf()) my_error("Can't go to a point if not localized");
    /*  if ( Mem->DistanceTo(p) < buffer && Mem->BallPositionValid() ){    //LPR
BallPositionValid!    if ( Mem->SP_use_offside && fabs(Mem->MyX() - Mem-
>my_offside_line) < 5 ){   // hack       Unum opp = Mem-
>FurthestForwardOpponent();      if ( opp != Unum_Unknown && Mem-
>OpponentPositionValid(opp) < .9 ){        // hack  Mem->LogAction2(40,
"go_to_point: looking for offsides line");   return face_neck_to_opponent(opp);
}    }    Mem->LogAction2(40, "go_to_point: already at the point");    return
AQ_ActionNotQueued;  }*/
    if (Mem - > PlayMode == PM_Their_Goal_Kick && Mem - > MyPos() != p) {
```

```
        /* if ( Mem->TheirPenaltyArea.IsWithin(p) ){      my_error("Can't go into
their penalty area on a goal kick!"); */
        Line l = LineFromTwoPoints(Mem - > MyPos(), p);
        Vector intersection = AdjustPtToRectOnLine(Mem - > MyPos(), Mem - >
TheirPenaltyArea, l);
        if (intersection != Mem - > MyPos() && l.InBetween(intersection, Mem - >
MyPos(), p)) { /* Need to go around the rectangle */
            Mem - > LogAction2(40, "go_to_point: moving around penalty area");
            Vector target;
            if (Mem - > MyX() < Mem - > TheirPenaltyArea.LeftX()) target =
Vector(Mem - > TheirPenaltyArea.LeftX() - 3, p.y);
            else if (Mem - > MyY() > 0) target = Vector(Mem - >
TheirPenaltyArea.LeftX() - 3, Mem - > TheirPenaltyArea.BottomY() + 3);
            else target = Vector(Mem - > TheirPenaltyArea.LeftX() - 3, Mem - >
TheirPenaltyArea.TopY() - 3);
            go_to_point(target, 0, dash_power, dodge);
            return AQ_ActionQueued;
        }
    }
    if (Mem - > PlayMode != PM_Play_On && Mem - > TeamInPossession() == Mem - >
TheirSide && !Mem - > OffsidePosition(p, Mem - > MySide) && //p.dist(Mem-
>BallAbsolutePosition()) > Mem->SP_free_kick_buffer &&
        Mem - > InOffsidePosition() && Mem - > BallDistance() < Mem - >
SP_free_kick_buffer + 1) {
        Mem - > LogAction2(40, "go_to_point: moving around free_kick area");
        if (Mem - > BallY() > Mem - > MyY()) go_to_point(Vector(Mem - > MyX(), Mem
- > BallY() - (Mem - > SP_free_kick_buffer + 1)));
        else go_to_point(Vector(Mem - > MyX(), Mem - > BallY() + (Mem - >
SP_free_kick_buffer + 1)));
        return AQ_ActionQueued;
    }
    float target_ang = GetNormalizeAngleDeg((p - Mem - >
MyPredictedPosition()).dir() - Mem - > MyBodyAng());
    float target_dist = Mem - > DistanceTo(p);
    if (dodge != DT_none) { /* dodge players */
        PlayerObject * player;
        float dodge_dist = Min(Mem - > CP_dodge_distance_buffer, target_dist);
        AngleDeg dodge_ang = Mem - > CP_dodge_angle_buffer;
        if ((player = Mem - > GetPlayerWithin(dodge_dist, dodge_ang, 0, target_ang
- dodge_ang)) != NULL && (dodge != DT_unless_with_ball || (Mem - >
BallPositionValid() && player - > get_abs_pos().dist(Mem - >
BallAbsolutePosition()) > Mem - > SP_kickable_area)) && (dodge !=
DT_only_with_ball || (Mem - > BallPositionValid() && player - >
get_abs_pos().dist(Mem - > BallAbsolutePosition()) <= Mem - > SP_kickable_area)))
{
            Mem - > LogAction2(40, "go_to_point: dodging right"); /*if ( Mem-
>NumPlayersWithin( dodge_dist, 2*dodge_ang) ){*/ /* Target at dist player_size, so
no players will be within in the next iteration ==> dash */
            Vector new_target = Mem - > BodyPolar2Gpos(Mem - > SP_player_size,
player - > get_ang_from_body() + Mem - > CP_dodge_angle);
            if (new_target == p) my_error("Dodging isn't changing the point!");
            go_to_point(new_target, 0, Mem - > CP_dodge_power, DT_none);
            /*}      else{ dash(Mem-
>CorrectDashPowerForStamina(Min(dash_power,Mem->CP_dodge_power))); }*/
            return AQ_ActionQueued;
        }
        if ((player = Mem - > GetPlayerWithin(dodge_dist, dodge_ang, 0, target_ang
+ dodge_ang)) != NULL && (dodge != DT_unless_with_ball || (Mem - >
BallPositionValid() && player - > get_abs_pos().dist(Mem - >
```

```
BallAbsolutePosition()) > Mem - > SP_kickable_area)) && (dodge !=
DT_only_with_ball || (Mem - > BallPositionValid() && player - >
get_abs_pos().dist(Mem - > BallAbsolutePosition()) <= Mem - > SP_kickable_area)))
{
            Mem - > LogAction2(40, "go_to_point: dodging left"); /*if ( Mem-
>NumPlayersWithin( dodge_dist, 2*dodge_ang) ){*/ /* Target at dist player_size, so
no players will be within in the next iteration ==> dash */
            Vector new_target = Mem - > BodyPolar2Gpos(Mem - > SP_player_size,
player - > get_ang_from_body() - Mem - > CP_dodge_angle);
            if (new_target == p) my_error("Dodging isn't changing the point!");
            go_to_point(new_target, 0, Mem - > CP_dodge_power, DT_none);
            /*}      else{  dash(Mem-
>CorrectDashPowerForStamina(Min(dash_power,Mem->CP_dodge_power)));      }*/
            return AQ_ActionQueued;
        }
    }
    if (fabs(target_ang) > Mem - > CP_max_go_to_point_angle_err || (Mem - >
PlayMode == PM_Their_Goal_Kick && Mem - > TheirPenaltyArea.IsWithin(Mem - >
MyPredictedPosition(1, dash_power)))) {
        Mem - > LogAction3(50, "go_to_point: turning %f", target_ang);
        turn(target_ang);
        return AQ_ActionQueued;
    } // LPR - Vou tirar isto //  dash_power = Mem-
>CorrectDashPowerForStamina(dash_power);
    if (dash_power > 0) {
        Mem - > LogAction3(50, "go_to_point: dashing %f", dash_power);
        dash(dash_power);
        return AQ_ActionQueued;
    } else {
        my_stamp;
        printf("recovering\n");
    }
    /*  if ( Mem->DistanceTo(p) < buffer && !Mem->BallPositionValid() )
test_face_ball();  // LPR*/
    Mem - > LogAction2(50, "go_to_point: doing nothing?");
    return AQ_ActionNotQueued;
}
```

Tím magmaOffenburg taktiež podľa prístupných kódov nepoužíva euklidovské transformácie.

V kóde sme našli iba výpočty zmien videnia lopty pri behu.

```
@
Override
protected RunInformation createRunInformation(Random rand, int runID) {
    double radius = (runID + 3);
    double angle = Math.toRadians(noise(rand, BEAM_ANGLE));
    double ballX = -radius * Math.cos(angle);
    double ballY = radius * Math.sin(angle);
    double beamX = ballX - DISTANCE_BEHIND_BALL + noise(rand, BEAM_NOISE);
    double beamY = ballY + noise(rand, BEAM_NOISE);
    return new RunInformation(runID, beamX, beamY, ballX, ballY);
}
```

5.1.5. Analýza Euklidovských matíc a transformácií

Euklidovské transformácie sú vhodné na menenie polohy v priestore, zachovávajú veľkosť útvaru. Matica lineárnej transformácie má tvar:

$$(x, y, z, 1) \rightarrow (x', y', z', 1)$$

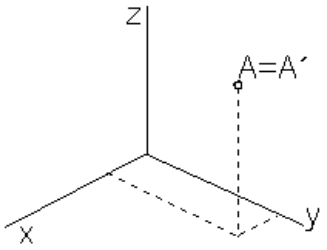
$$x' = f(x, y, z),$$

$$y' = g(x, y, z),$$

$$z' = h(x, y, z)$$

$$A' = A \,.T$$

Identita, základný stav:

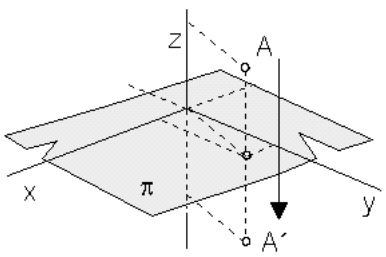


$$\mathbf{T}_I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} x' = x \\ y' = y \\ z' = z \\ \mathbf{A' = A.T_I} \end{matrix}$$

Obrázok 16 Identita v matici

5.1.5.1. Základné transformácie:

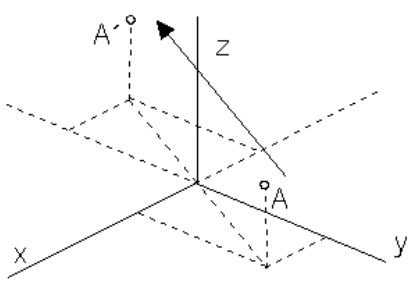
Rovinná súmernosť podľa súradnicovej roviny $p = xy$:



$$\mathbf{T}_{S_\pi} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} x' = x \\ y' = y \\ z' = -z \\ \mathbf{A' = A.T_{Sp}} \end{matrix}$$

Obrázok 17 Rovinná súmernosť v matici

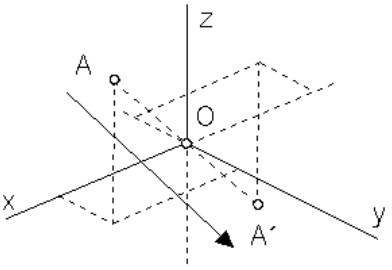
Osová súmernosť podľa súradnicovej osi z:



$$\mathbf{T}_{Sz} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} x' = -x \\ y' = -y \\ z' = z \\ \mathbf{A' = A.T_{Sz}} \end{matrix}$$

Obrázok 18 Osová súmernosť v matici

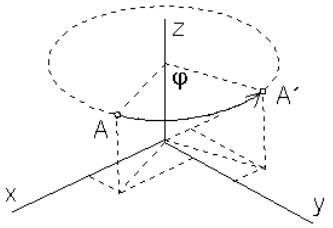
Stredová súmernosť podľa začiatku súradnicovej sústavy *O*:



$$\mathbf{T}_{SO} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} x' = -x \\ y' = -y \\ z' = -z \\ \mathbf{A' = A.T_{SO}} \end{matrix}$$

Obrázok 19 Stredová súmernosť v matici

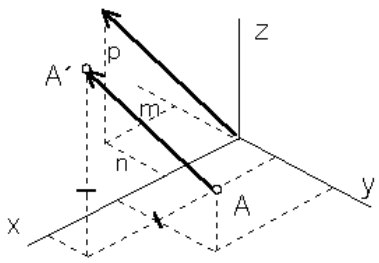
Otáčanie okolo osi *z* o uhol *j*:



$$T_{Oz} = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 & 0 \\ -\sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} x' = x\,\mathrm{cos}j - y\,\mathrm{sin}j \\ y' = x\,\mathrm{sin}j + y\,\mathrm{cos}j \\ z' = z \\ \mathbf{A' = A.T_{Oz}} \end{matrix}$$

Obrázok 20 Otáčanie okolo osi v matici

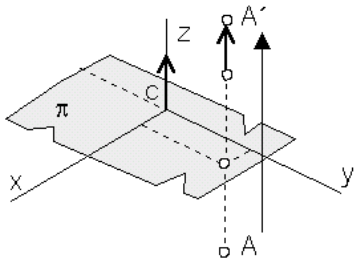
Posunutie o vektor (*m*, *n*, *p*, *0*):



$$\mathbf{T}_{PS_\pi} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & c & 0 & 1 \end{pmatrix} \begin{matrix} x' = x + m \\ y' = y + n \\ z' = z + p \\ \mathbf{A'} = \mathbf{A} . \mathbf{T_P} \end{matrix}$$

Obrázok 21 Posunutie o vektor v matici

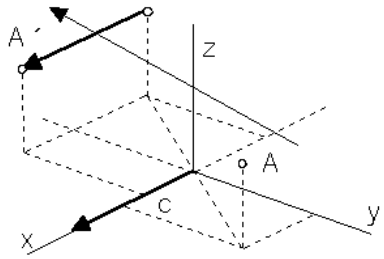
Posunutá rovinová súmernosť podľa súradnicovej roviny *p* s posunutím o vektor *(0, c, 0, 0)*:



$$\mathbf{T}_{PS_\pi} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & c & 0 & 1 \end{pmatrix} \begin{matrix} x' = x \\ y' = y \\ z' = -z + c \\ \mathbf{A'} = \mathbf{A} . \mathbf{T_{PSp}} \end{matrix}$$

Obrázok 22 Posunutá rovninová súmernosť v matici

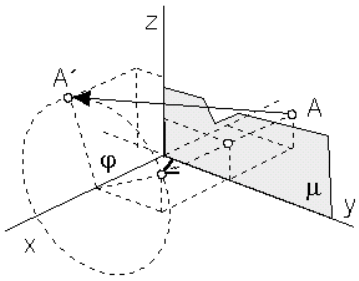
Posunutá osová súmernosť podľa súradnicovej osi *z* s posunutím o vektor (*c, 0, 0, 0*):



$$\mathbf{T}_{PS_z} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ c & 0 & 0 & 1 \end{pmatrix} \begin{matrix} x' = -x + c \\ y' = -y \\ z' = z \\ \mathbf{A'} = \mathbf{A} . \mathbf{T_{PSz}} \end{matrix}$$

Obrázok 23 Posunutá osová súmernosť v matici

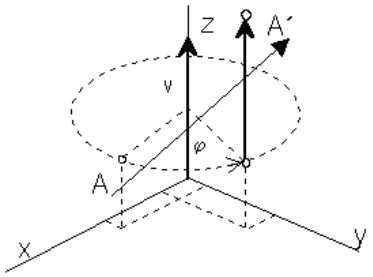
Otočená rovinová súmernosť podľa súradnicovej roviny *m* okolo súradnicovej osi *x*:



$$\mathbf{T}_{OSv} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi & 0 \\ 0 & -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} x' = -x \\ y' = y\cos j - z\sin j \\ z' = y\sin j + z\cos j \\ A' = A.T_{OSn} \end{matrix}$$

Obrázok 24 Otočená rovinová súmernosť v matici

Skrutkový pohyb okolo osi *z* s posunutím o vektor (*0,0,v,0*) pre uhol otočenia *j*:



$$\mathbf{T}_S = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 & 0 \\ -\sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & v & 1 \end{pmatrix} \begin{matrix} x' = x\cos j - y\sin j \\ y' = x\sin j + y\cos j \\ z' = z + v \\ A' = A.T_S \end{matrix}$$

Obrázok 25 Skrutkový pohyb v matici

5.1.5.2. Afinné transformácie

Afinné transformácie nezachovávajú veľkosti úsečiek a uhlov. Každá euklidovská transformácia je afinnou transformáciou (podobnosťou) v ${}_{\infty}E^3$.

Všeobecná afinná transformácia:

$$\mathbf{T}_A = \begin{pmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ m & n & p & s \end{pmatrix}$$

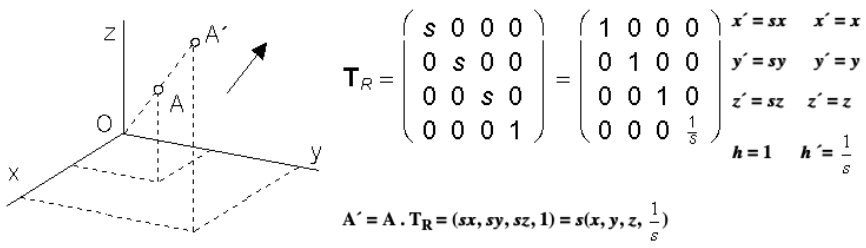
$|\mathbf{T}_A| \neq 0$, je maticou sústavy rovníc

$$\begin{aligned} x' &= ax + dy + gz + m \\ y' &= bx + ey + hz + n \\ z' &= cx + fy + iz + p \\ h' &= s \end{aligned}$$

$$A = (x, y, z, 1) \to A' = (x', y', z', h') = A.\mathbf{T}_A = s\left(\tfrac{x'}{s}, \tfrac{y'}{s}, \tfrac{z'}{s}, 1\right)$$

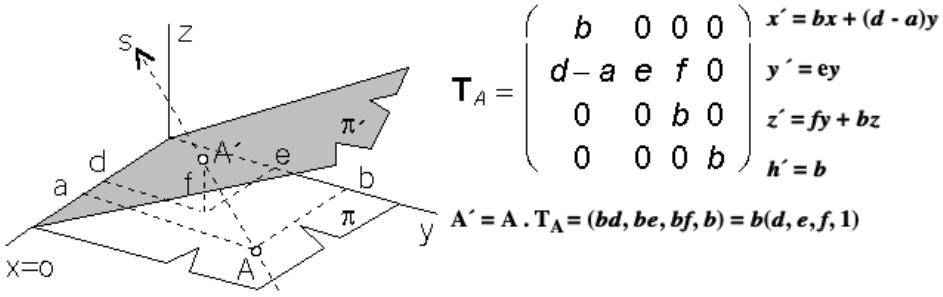
Obrázok 26 Afinná transformácia

Rovnoľahlosť s daným stredom O a nenulovým koeficientom rovnoľahlosti s (zmena mierky):



Obrázok 27 Rovnoľahlosť v matici

Osová afinita medzi rovinou p = xy a rovinou p´ = xA´ s osou v osi x a odpovedajúcou si dvojicou bodov A(a, b, 0, 1) ® A'(d, e, f, 1), b 0, f 0, so smerom afinity určeným priamkou s = AA' so smerovým vektorom $\bar{s}$



Obrázok 28 Osová afinita v matici

Zdrojový kód:

1. http://zetcode.com/gfx/java2d/transformations/
2. http://www.java2s.com/Code/Java/2D-Graphics-GUI/Variousgeometrictransformationsonmatrixform.htm

5.1.6. Analýza kalmanovho filtra

Kalmanov filter je algoritmus, ktorý používa sériu meraní pozorovaných v priebehu času, obsahujúcich štatistický šum a iné nepresnosti na odhad premenných v širokom spektre procesov. Je široko používaný pre spracovanie signálov, navigáciu a iné úlohy. Predpoklad pre využitie kalmanovho filtra je linearizovaný systém, alebo systém transformovaný na lineárny.

V projekte sa kalmanov filter používa na redukciu šumu, a teda presnejšie určenie pozície objektov, ktoré hráč vidí, teda pozície lopty a pozície pevných bodov.

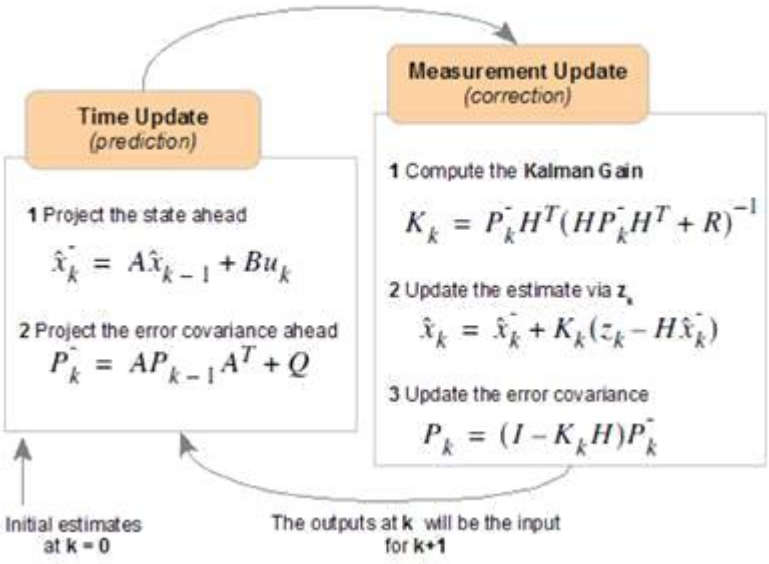
5.1.6.1. Lineárne systémy:

V uvedených rovniciach pre lineárne systémy platí:

1. A,B a H sú matice
2. k je časový index
3. x je stav systému
    1. vektor x obsahuje všetky dostupné informácie o systéme
    2. nie je možné ho merať
4. P je matica - predikovaná kovariancia stavu
5. K je matica - optimálny kalman zisk
6. u je známy vstup
7. w je procesný šum
8. z je merací šum

5.1.6.2. Rovnice kalmanovho filtra

Kalmanov filter sa delí na 2 fázy: Predikčnú, počas ktorej sa robí odhad ako predikcia z predchádzajúceho najlepšieho odhadu plus korekcia. Táto fáza obsahuje dve rovnice. Druhá fáza je korekčná fáza, ktorá obsahuje tri rovnice. Ich tvar môžeme vidieť na nasledujúcom obrázku.



Obrázok 29 Vzťah rovníc kalmanovho filtra

5.1.6.3. Starý kalmanov filter v projekte

Tento kalman bol implementovaný v roku 2010 tímom Androids. Využívajú sa v ňom triedy KalmanForVector, ktorý je počítaný pomocou triedy KalmanForVariable aplikovanej na všetky súradnice vektora.

- KalmanForVariable slúži na výpočet linearno-kvadratického Gaussovho regulátora pre premennú - v tomto prípade súradnicu
- KalmanForVector slúži na aplikovanie kalmanovho filtra na vektor
- KalmanAdjuster je trieda slúžiaca na výpočet polohy objektov na ihrisku použitím vyššie uvedených tried

Tento kalmanov filter nie je optimálny, a preto sa ho v roku 2017 pokúsil nahradiť tím BAREKO.

5.1.6.4. Nový kalmanov filter v projekte

Kalmanov filter je implementovaný v triede KalmanReal, v balíku sk.fiit.jim.agent.kalman. Tento kalmanov filter využíva matice a vektory definované v teoretickej časti a rovnice implementuje podľa teórie kalmanovho filtra. V triede sa nachádzajú dve hlavné metódy:

- Public void predict()
    - táto metóda obsahuje rovnice prvej predikčnej fázy kalmanovho filtra
- public void update(RealVector z)
    - táto metóda obsahuje rovnice z druhej korekčnej fázy kalmanovho filtra

Tento kalmanov filter je tiež volaný v triede KalmanAdjuster. Matice a vektory sa tiež inicializujú v tejto triede. Ich hodnoty sa nedajú presne overiť, keďže hodnoty v maticiach sú unikátne pre každý problém, preto ich tím BAREKO testoval na základe im dostupných informácií a metódou pokus - omyl. Podrobné informácie o testovaní hodnôt matíc sa nachádza v inžinierskom diele tímu BAREKO.

5.1.6.5. Navrhnutá optimalizácia

Nový kalman sa v projekte využíva iba na upravovanie pozície lopty. Upravovanie pozície pevných bodov prebieha stále pomocou starého kalmana, ktorý dosahuje horšie výsledky ako nový kalman implementovaný pomocou matíc. Preto hlavnou navrhnutou optimalizáciou bude aplikovanie nového kalmana aj na pevné body, upravenie hodnôt matíc a vektorov pre tieto pevné body, a následné testovanie. Ďalším problémom je opakovaná inicializácia nového

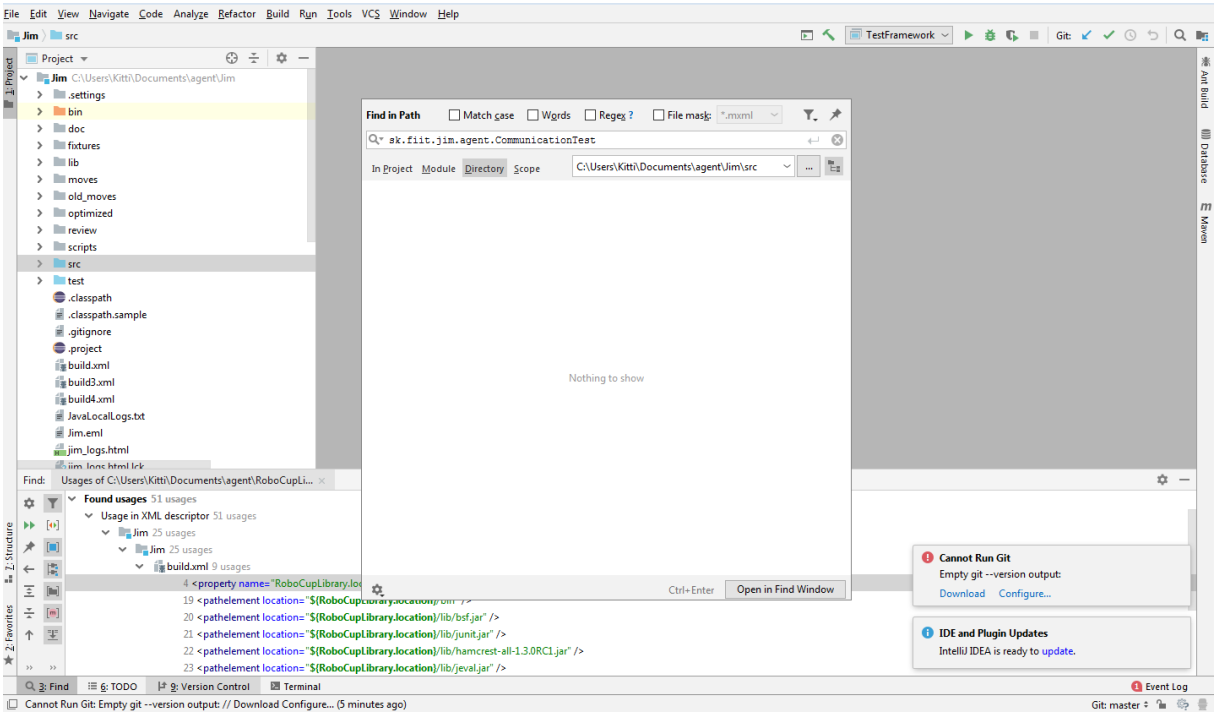
kalmana, ktorý sa inicializuje pred každým výpočtom a tým pádom sa vypočítané hodnoty nepoužívajú v ďalších krokoch. Túto opakovanú inicializáciu môžeme vidieť v nasledujúcej ukážke z kódu, kde sa pri každom spracovávaní správy inicializujú hodnoty matíc.

```
*/
public void processNewServerMessage(ParsedData data) {
    initMatrixes();
    this.now = data.SIMULATION_TIME;
    adjustBallPosition(data);
    adjustFixedPointsPosition(data.fixedObjects);
}
```

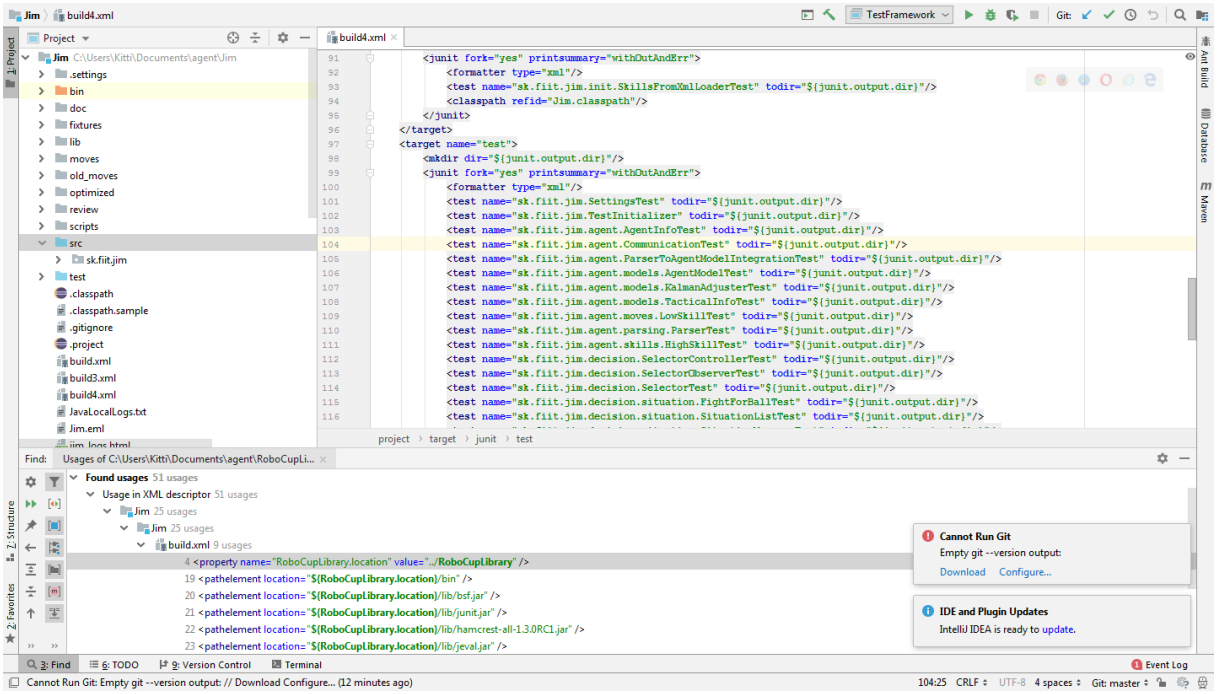
Obrázok 30 Metóda processNewServerMessage

5.1.7. Analýza nepoužívanej triedy CommunicationTest

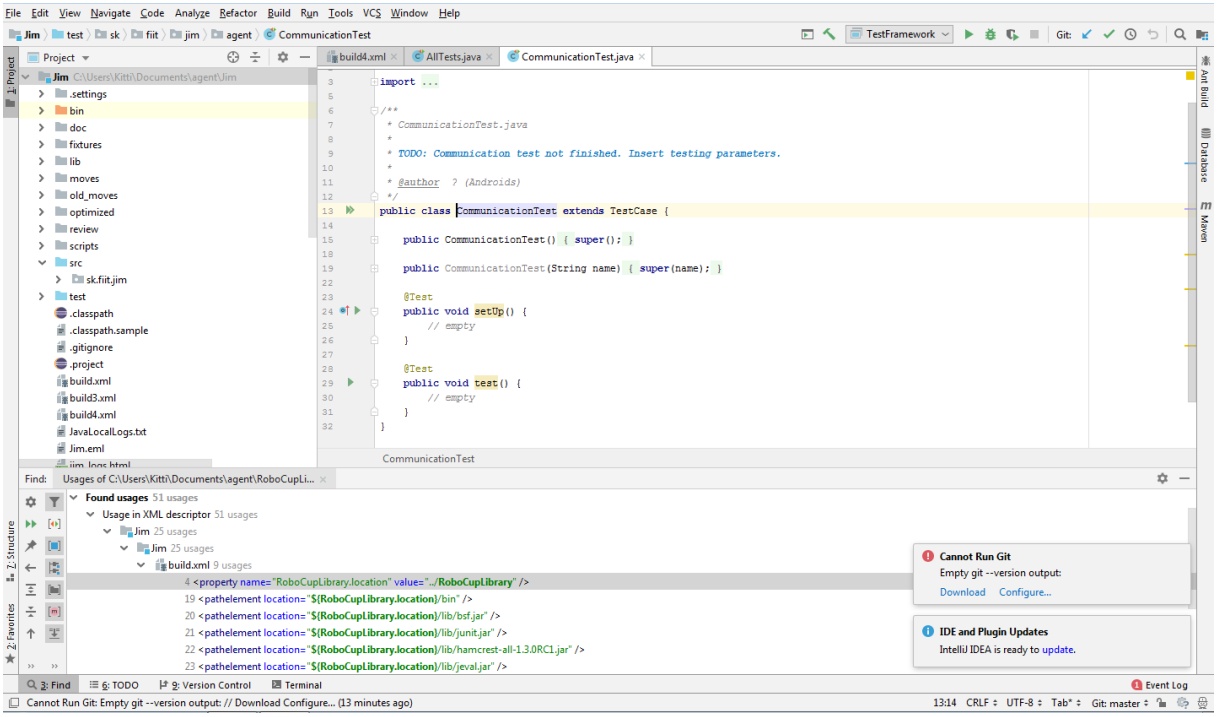
V projekte sa našli 2 miesta, kde sa daná trieda vyskytuje. Jedno je miesto, kde sú zapísané všetky testy, ktoré sa v projekte nachádzajú (2. obrázok v poradí). Druhé miesto, kde sa daný test vyskytuje je samotná implementácia triedy, ktorá je prázdna. Po analýzy dostupnej dokumentácie sme nezistili, ako pôvodný tím myslel implementáciu tejto triedy.



Obrázok 31 Hľadanie triedy v projekte



Obrázok 32 Výskyt triedy v projekte 1



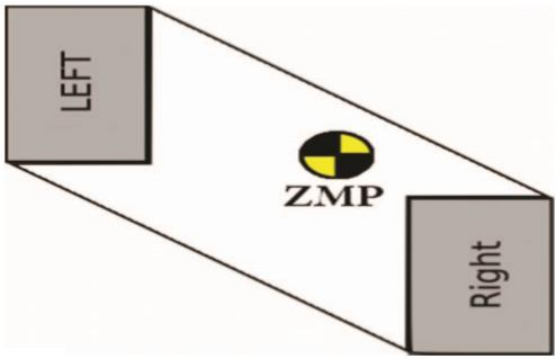
Obrázok 33 Výskyt triedy v projekte 2

Z dostupnej dokumentácie nie je jasné, ako tím pôvodne myslel implementáciu danej triedy. Po dohode s vedúcim sme danú triedu zmazali.

5.1.8. Zero moment point Jima

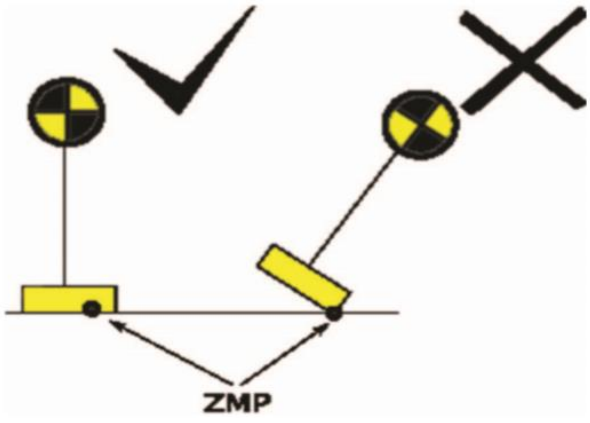
Zero moment point (ZMP) berie v úvahu efekty dynamických a statických síl. Špecifikuje bod, v ktorom dynamická reakcia síl v kontakte s chodidlom a zemou neprodukujú žiadny moment v horizontálnom smere. Vo väčšine prípadov je za ZMP považovaný bod, kde celkový horizontálny vplyv a gravitačné sily sú rovné nule.

Stabilita počas kráčania vyžaduje ZMP vnútri podporného polygónu. Podporný polygón je vykreslený ako konvexný trup v okolí chodila a zeme.



Obrázok 34 ZMP polygón

Udržanie ZMP v podpornom polygón zabezpečuje stabilitu počas kráčania. V prípade, že sa ZMP dostane mimo podporného polygónu, veľmi rýchlo môže nastať jeho pád.



Obrázok 35 Strata rovnováhy

5.1.8.1. Identifikácia v kóde

ZMP by sa mal v kóde vyskytovať hlavne v agentovi – highskill – move. Pri WalkFast sú vytvorené 3 súbory s názvom WaklFast, WaklFastZMP a WaklFastZMPOld. Podľa dokumentácie priamo v kóde ťažko určiť, ktorá z týchto tried je využívaná. V kóde je taktiež veľa zapoznámkovaných časti kódu, ku ktorým opäť často chýba popis a ťažko tak určiť dôvod

zapoznámkovania. Nakoľko trieda WalkFast neobsahuje žiadne prvky stabilizácie je možné predpokladať, že ZMP nebol v Jimovi riešený vôbec. Naopak trieda WalkFastZMP obsahuje len stabilizačné metódy. Táto trieda je importovaná iba triedou DefaultTactic. WalkFastZMPOld je importovaná pri FastWalkTactic a StabilityWalkTactic. Táto trieda je aj zapoznámkovaná narozdiel od WalkFastZMP. V triede FastWalkTactic je možné pomocou boolean hodnoty určiť, či bude ZMP spustený – defaultná hodnota je TRUE. V tomto prípade je na prepočet používaná trieda WalkFastZMPOld.

AgentModel obsahuje hlavnú prácu so ZMP, kde sa updatujú podstatné metódy ako updateZeroMomentPoint(), updateCenterOfMass(), updateFeetForce(), atď.

5.1.8.1.1. **sk.fiit.jim.agent.models.AgentModel**

Táto trieda je najpodstatnejšia pre ZMP, nakoľko v nej prebiehajú všetky výpočty spojené so ZMP. Obsahuje 3 podstatné metódy: updateZeroMomentPoint(), updateCenterOfMass() a setZMPallowed(boolean zmpAllowed).

5.1.8.1.2. **updateZeroMomentPoint()**

Táto metóda počíta samotný ZMP. Síce sa na začiatku vytvoria premenné x, y, z s prednastavenou hodnotou 0, no z súradnica sa počas výpočtov vôbec nemení. Už v 2. riadku tejto metódy vidíme TODO na opravu stredu accelerometra. Vzorec použitý na výpočet xzmp a yzmp je:

$$x_{ZMP} = -\left(x_{com} - \frac{z_{COM} \cdot x_{ACC}}{z_{momentum}}\right) \qquad y_{ZMP} = -\left(y_{com} - \frac{z_{COM} \cdot y_{ACC}}{z_{momentum}}\right)$$

Potom sa pomocou karteziánskeho súčinu vzniknutých x,y a z vypočíta hodnota ZMP a tá sa zapíše do LinkedListu obsahujúceho históriu vypočítaných ZMP. Následne sa ako ZMP vypočíta priemer posledných 5 vypočítaných ZMP hodnôt.

5.1.8.1.3. **updateCenterOfMass()**

Na výpočet je použitý vzorec:

$$COM = \frac{\sum_{i=0}^{n} p_i \, m_i}{\sum_{i=0}^{n} m_i},$$

Kde n je počet súčiastok robota, p je poloha súčiastky, m je hmotnosť súčiastky. V cykle sa pre každú súčiastku pridáva do 3D vektora centerOfMass súčin pozície súčiastky a jej váha. Do

celkovej hmotnosti totalMass sa postupne pripočítava hmotnosť aktuálnej súčiastky. Nakoniec sa vektor centerOfMass vydelí totalMass a vznikne z toho výsledný COM.

5.1.8.1.4. **setZMPallowed(boolean zmpAllowed)**

Táto trieda má za účel zapnutie a vypnutie ZMP. Taktiež prispôsobí minimálne hodnoty Thety na prednastavené hodnoty. Zaujímavé je, že hodnoty Theta sa využívajú len pri zisťovaní pádu a sú rozličné pri zapnutom a vypnutom ZMP. Taktiež nie je žiadne volanie, ktoré by zisťovalo či je ZMP zapnuté alebo nie. Túto metódu volajú s parametrom true dve triedy v balíčku sk.fiit.jim.agent.highskill.move WalkFastZMP a WalkFastZMPOld. Triedu WalkFastZMP využíva taktika DefaultTactic a triedu WalkFastZMPOld využívajú taktiky FastWalkTactic a StabilityWalkTactic.

Metóda updateZeroMomentPoint() vôbec nemení premennú z počas výpočtu a necháva ju rovnú 0 – preto je dôležité overiť a prípadne opraviť tento výpočet. Taktiež obsahuje TODO časť v komentári, ktorú poznačuje na možný nesprávny výpočet ZMP. Vo výpočte je používaný LinkedList a je prehľadávaný cyklom for – tento cyklus je neefektívny pre prehľadávanie LinkedListu a odporúčam optimalizovanie pomocou foreachu. Do histórie sa počas behu tejto metódy pridáva jeden prvok, preto by bolo vhodné namiesto while cyklu použiť jednoduchú if podmienku pri zisťovaní prekročenia maximálnej veľkosti a následnom vyprázdňovaní prvkov z LinkedListu. Maximálna veľkosť LinkedListu je nastavená na 1000 prvkov, no pracujeme len s 5-timi, preto je zbytočné udržiavať ostatných 955 prvkov v zozname.

5.1.9. UDP/TCP komunikácia so serverom u iných hráčov

Náš tím analyzoval ako jeden z ťažiskových problémov nedostatočnú rýchlosť spracovania operácií Jima, resp. ich príliš veľké množstvo. Ako prvý nápad bolo analyzovanie preposielanie informácií medzi Jimom a serverom. Naše predpoklady, že UDP protokol je výhodnejší z dôvodu výrazne vyššej rýchlosti a menšej veľkosti packetov(8 vs 20 bit hlavička) boli potvrdené krátkou analýzou tohto riešenie u iných tímov. V našom tíme bude nasledovať kontrola kódu Jima, či naozaj využíva výhradne UDP protokol.

5.1.9.1. Riešenie tímu Austin Villa

Ako najlepší tím sme pracovali v prvom rade s ich výskumom. Veľmi stručne je v ich výskume rozhodnutie pre UDP, nakoľko komunikácia v intervale 20 - 25 ms nepotrebuje maximálnu spoľahlivosť ale predovšetkým maximálnu rýchlosť.

“UDP One of our primary decisions was to decide between using the Transmission Control Protocol(TCP) and the User Datagram Protocol(UDP). While TCP is more reliable, we determined that the additional overhead associated with using TCP was not worth the reliability benefits it bestowed. Meanwhile, a convenient feature of UDP is the ability to broadcast without assembling and sending multiple messages.“

https://www.cs.utexas.edu/users/AustinVilla/papers/Villa05-legged-tech.pdf [1]

5.1.9.2. Riešenie tímu Androids FIIT

Opis riešenia z dokumentáciu Tímu Androids – FIIT 2010/2011[2] navrhuje použitie oboch spomínaných protokolov a vytvorenie TCP aj UDP socketu. Po preskúmaní kódu Jima 2018 však bol nájdený už len UDP resp. TFTP socket z čoho vyplýva, že TCP bolo niekedy v priebehu rokov vypustené z projektu úplne.

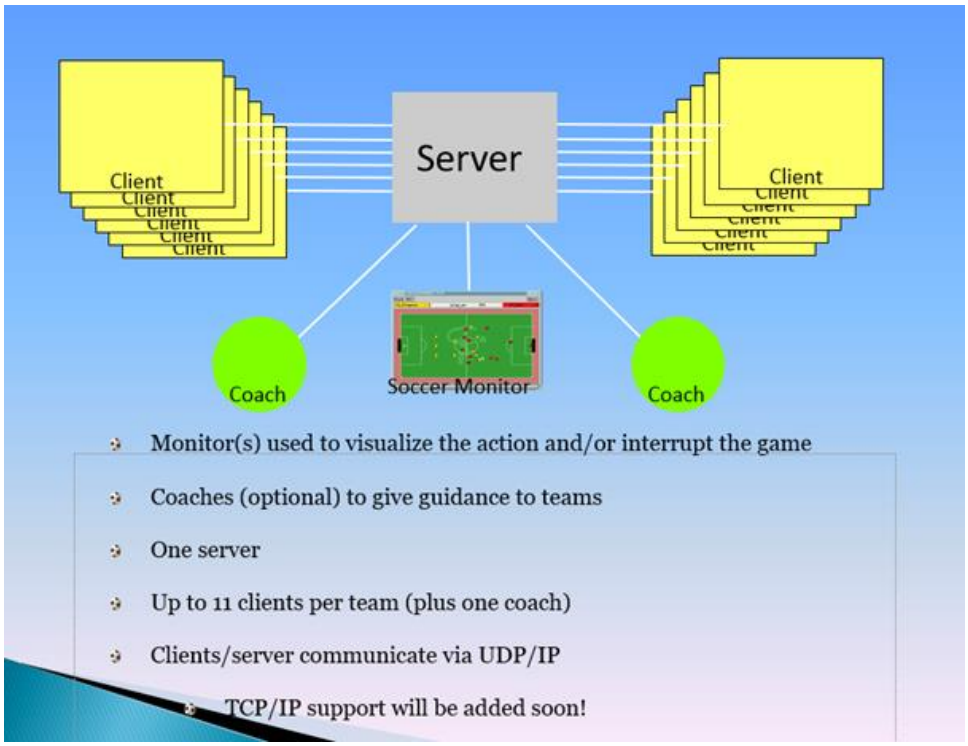
“Keďže komunikácia prebieha pomocou TCP aj UDP protokolov, pre agenta boli implementované TCP a UDP sokety – využité boli POSIX sokety. „

5.1.9.3. Riešenie tímu RoboCup-99: Robot Soccer World Cup III

Spracovanie a posielanie údajov cez UDP socket bolo dokonca zakomponované do Third Robot World Cup Soccer Games and Conferences, Robo-Cup-99, ktorá sa konala v Štokholme, Švédsko v júl/august 1999. Táto kniha spracúvala vtedajšie poznatky a výskum najlepších tímov zúčastnených na RoboCup 1999 v Štokholme.

5.1.9.4. Riešenie tímu US Penn

Ako je vidieť na snímku prezentácie jedného z popredných tímov v súťaži RoboCup – US Pennalizers, na komunikáciu so serverom rovnako používajú UDP sockety, TCP spomínajú ako možnú implementáciu v roku 2009, podľa ich aktuálne zverejneného kódu v roku 2018 však implementovaná nie je.



Obrázok 36 Komunikácia so serverom tímu US Penn

5.1.9.5. Simulačný server SimSpark

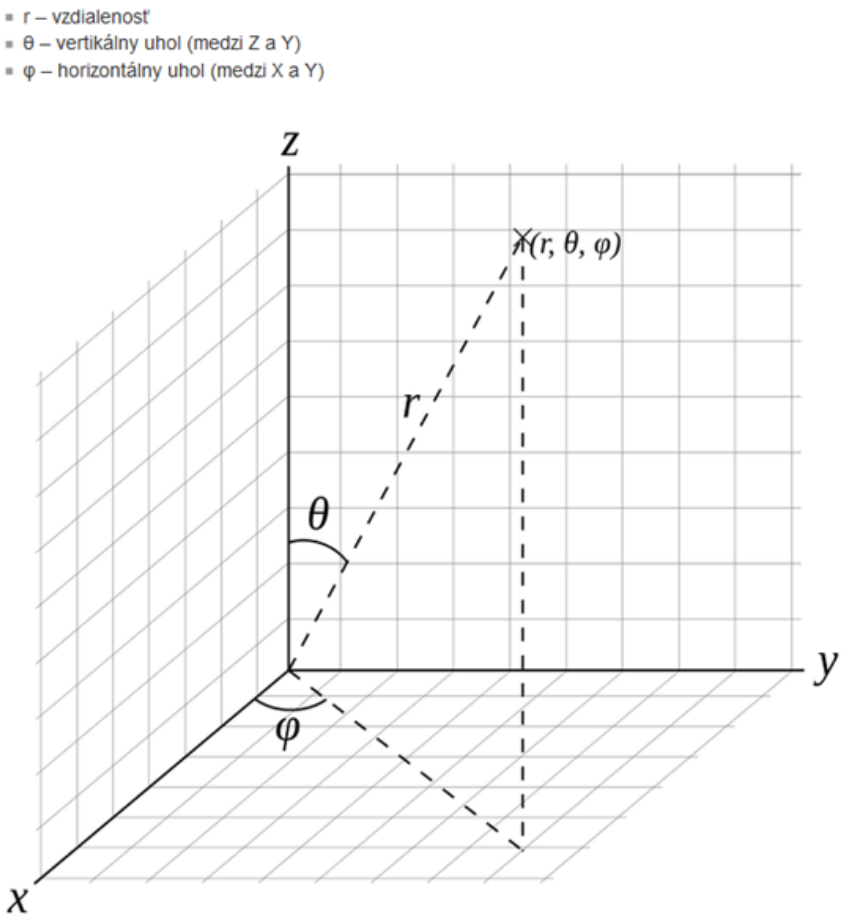
Komunikácia hráčov so serverom SimSpark prebieha údajne na porte 3200 nad protokolom TCP a to aj podľa SimSpark wiki http://simspark.sourceforge.net/wiki/index.php/Network_Protocol. Tento údaj však neprešiel písomnou aktualizáciou od roku 2008. Podľa novších informácií SimSpark simulačný server podporuje aj UDP spojenie.

Zo všetkých spomínaných zdrojov je zjavné, že využitie UDP má svoje výhody oproti TCP a to najmä rýchlosť spracovania. Z týchto dôvodov sme si komunikáciu nad protokolom UDP zvolili aj my pre tím WAWOT v roku 2018-2019. Nakoľko sme z kódu Jima zistili, že toto riešenie už implementované je, nasledovať bude už len krátka analýza samotného využitia TFTP socketu v aplikácií.

5.1.10. Kontrola osí x/y

Pri rôznych pokusoch implementácií v testovacom režime TestFramework v projekte bola identifikovaná výmena osí X a Y od plánu projektu a bežnej konvencie označovania osí – X horizontálna os, Y vertikálna os. Táto výmena osí v základných výpočtoch projektu nebola plánovaná a značne sťažovala prácu pri návrhu základných pohybov a funkcií pre pohyb a orientáciu hráča Jima v priestore.

Analýzou fungovania samotného vnímania polohy a prepočítavania vzdialeností zo základných údajov od servera sme dostali predstavu o správnom modeli týchto výpočtov. Náčrt osí a uhlov použitých pri výpočtoch sa nachádza aj na tímovej wiki.



Obrázok 37 Popis osí a uhlov v prostredí simulácie

Vychádzajúc z tejto schémy a rozmiestnenia uhlov phí a theta sme skontrolovali výpočty týchto osí v triede Vector3D nasledovne:

\RoboCupLibrary\src\sk\fiit\robocup\library\geometry\Vector3D.java

```
/*
 * Computes spherical coordinates from cartesian coordinates.
 */
private void calculateSpherical() {
   r = sqrt(x * x + y * y + z * z);
   theta = asin(z / r);
   if (r == 0.0)
      theta = 0.0;
   phi = atan2(y, x);
   phi -= PI / 2.0d;
```

```
   phi = Angles.normalize(phi);
   theta = Angles.normalize(theta);
}
```

Pričom bolo zistené, že uhly phi a theta sú vymenené vo výpočte. Pre porovnanie kód tímu UT Austin Villa:

Theta:

```
\return the direction in degrees of the vector corresponding with
the current VecPosition */

double VecPosition::getTheta( ) const {
return ( atan2Deg( m_y, m_x ) );
}
```

Phi:

```
    \return the direction in degrees of the vector corresponding with
    the current VecPosition */

double VecPosition::getPhi( ) const {
    return ( atan2Deg( m_z, sqrt(m_x * m_x + m_y * m_y)) );
}
```

Obdobne sú zamenené aj osi v ďalšej funkcii na výpočet karteziánskych súradníc:

```
/*
 * Computes cartesian coordinates from spherical coordinates.
 */
private void calculateCartesian() {
   x = cos(theta) * sin(phi) * -r;
   x = (int) (x * 1000.0) / 1000.0;
   y = cos(theta) * cos(phi) * r;
   y = (int) (y * 1000.0) / 1000.0;
   z = sin(theta) * r;
   z = (int) (z * 1000.0) / 1000.0;
}
```

UT Austin Villa:

```
VecPosition VecPosition::getVecPositionFromPolar(double r, double theta, double
phi) {
    // cos(phi) = x/r <=> x = r*cos(phi); sin(phi) = y/r <=> y = r*sin(phi)
    double x = r * cosDeg(phi) * cosDeg(theta);
    double y = r * cosDeg(phi) * sinDeg(theta);
    double z = r * sinDeg(phi);
```

```
return ( VecPosition(x, y, z) );
}
```

Ďalšie nezrovnalosti vo výpočtoch rotácií vektorov (rovnice nie sú správne v zmysle X,Y zamenené):

```
/**
 * Rotates vector over x coordinate of cartesian representation.
 *
* @param angleInRad
 *            angle to rotate
 * @return rotated vector
 */
public Vector3D rotateOverX(double angleInRad) {
   return cartesian(x, y * cos(angleInRad) - z * sin(angleInRad), y
         * sin(angleInRad) + z * cos(angleInRad));
}

/**
 * Rotates vector over y coordinate of cartesian representation.
 *
* @param angleInRad
 *            angle to rotate
 * @return rotated vector
 */
public Vector3D rotateOverY(double angleInRad) {
   return cartesian(x * cos(angleInRad) + z * sin(angleInRad), y, -x
         * sin(angleInRad) + z * cos(angleInRad));
}

/**
 * Rotates vector over z coordinate of cartesian representation.
 *
* @param angleInRad
 *            angle to rotate
 * @return rotated vector
 */
public Vector3D rotateOverZ(double angleInRad) {
   return cartesian(x * cos(angleInRad) - y * sin(angleInRad), y
         * cos(angleInRad) + x * sin(angleInRad), z);
}
```

UT Austin Villa:

```
VecPosition VecPosition::rotateAboutX(double angle) {
    double cos = cosDeg(angle);
    double sin = sinDeg(angle);
    double newX, newY, newZ;
    newX = m_x;
    newY = m_y * cos + m_z * sin;
    newZ = -m_y * sin + m_z * cos;
    return VecPosition(newX, newY, newZ);
}

VecPosition VecPosition::rotateAboutY(double angle) {
```

```
    double cos = cosDeg(angle);
    double sin = sinDeg(angle);
    double newX, newY, newZ;
    newX = m_x * cos - m_z * sin;
    newY = m_y;
    newZ = m_x * sin + m_z * cos;
    return VecPosition(newX, newY, newZ);
}

VecPosition VecPosition::rotateAboutZ(double angle) {
    double cos = cosDeg(angle);
    double sin = sinDeg(angle);
    double newX, newY, newZ;
    newX = m_x * cos + m_y * sin;
    newY = -m_x * sin + m_y * cos;
    newZ = m_z;
    return VecPosition(newX, newY, newZ);
}
```

Na základe porovnania nášho kódu a kódu UT Austin Villa sme dospeli k poznaniu, že vo výpočtoch uhlov a karteziánskych súradníc došlo k výmene súradníc X,Y a k zámene uhlov Phi a Theta. Identifikovaná zámena je v triede Vector3D. Podobný kód je zverejnený od UT Austin Villa v triede vecposition. Do budúcna sa navrhuje zapracovať tieto zmeny a otestovať ich vplyv na fungovanie ďalších funkcií programu.

Trieda Vector3D a jej výpočty sa používajú v nasledovných triedach, ktoré jej zmena ovplyvní:

bin/sk/fiit/jim/agent/models/AgentPositionCalculator.class

bin/sk/fiit/jim/agent/models/AgentRotationCalculator.class

bin/sk/fiit/jim/agent/models/AgentModel.class

bin/sk/fiit/jim/agent/models/AgentModelTest.class

bin/sk/fiit/jim/agent/models/BodyPart.class

bin/sk/fiit/jim/agent/models/DistanceHelper.class

bin/sk/fiit/jim/agent/models/FixedObject.class

bin/sk/fiit/jim/agent/models/PositionChangeGuardian.class

bin/sk/fiit/jim/agent/models/TacticalInfo.class

a množstvo ďalších.

5.1.11. Otočenie osí x/y

Analýzou fungovania samotného vnímania polohy a prepočítavania vzdialeností zo základných údajov od servera sme dostali predstavu o správnom modeli týchto výpočtov. Identifikovali sme chybné výpočty v triede sk.fiit.robocup.library.geometry.Vector3D.java a to konkrétne v porovnaní s implementáciou tímu UT Austin Villa. Zmeny sa opierali o analýzu z tasku 10002.

Prvým krokom bolo otočenie výpočtov v metódach:

spherical

calculateSpherical

calculateCartesian

rotateOverX

rotateOverY

rotateOverZ

Otočenie prinieslo očakávaný výsledok, a to konkrétne otočenie videnia hráča presne naopak. Čo spôsobilo neschopnosť akéhokoľvek koordinovaného pohybu. Podľa očakávaní bolo otočenie aj v iných metódach.

Analýze otočenia v iných triedach vyššej úrovne sa venoval task 10856. Task 10856 skončil s výsledkom, že ani otočenie v metódach pohybu neprinieslo očakávané výsledky a hráč naďalej nebol schopný koordinácie pohybov a videnia.

Zhodnotenie z tasku 10856:

Aby sme boli schopní chybu odstrániť, museli by sme od základov analyzovať celý kód a prekopať súčasné výpočty súradníc. Zhodli sme sa, že z dôvodu náročnosti v spojení s minimálnym prínosom (keďže všetka hráčova logika už je postavená s tým, že sú osi prehodené) sa nebudeme ďalej danej úlohe venovať.

Podrobnejšie sme analyzovali fungovanie funkcie rotateOver.

V porovnaní s https://github.com/LARG/utaustinvilla3d/blob/master/math/vecposition.cc

VecPosition VecPosition::crossProduct(VecPosition v)

je pravdepodobné otočenie práve v tejto funkcií.

- wiki a články ohľadom výpočtu sférických a karteziánskych súradníc sú neaktuálne a neúplné
- práca s maticami rotateOverX/Y/Z a funkcia rotateOver nie je zdokumentovaná

Ďalšou analýzou sme našli zdroj implementácie rotácie matíc v našom projekte.

https://en.wikipedia.org/wiki/Rotation_matrix (vzorce v časti Axis and angles)

Boli nájdené niektoré nezrovnalosti, ktoré boli opravené, avšak ich zmenou sa poškodila integrita fungovania robotického hráča a nebol schopný koordinovaného pohybu.

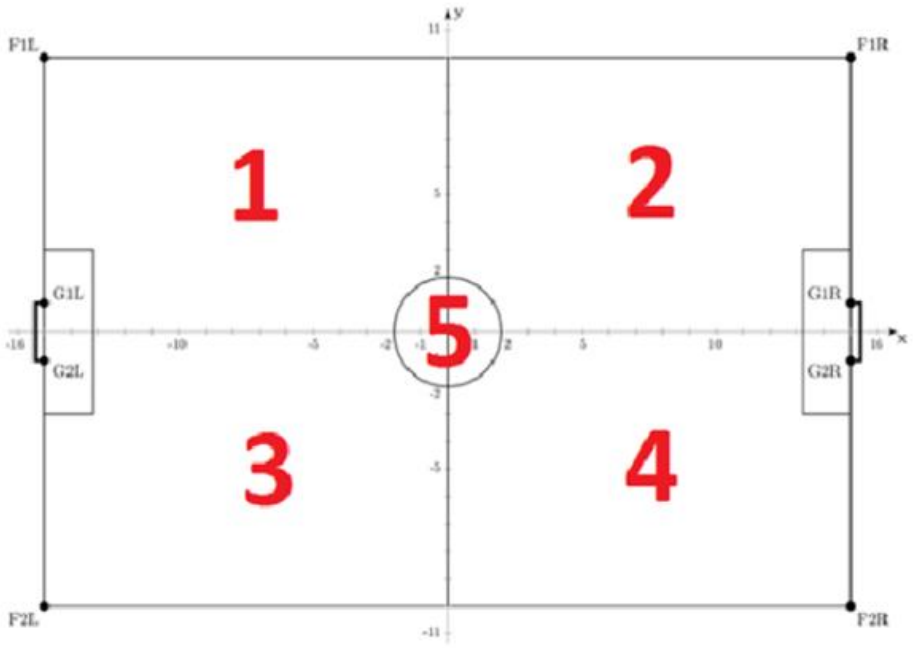
Ani jedna zo zmien nepriniesla želaný výsledok. Bez dokumentácie k implementácií danej triedy Vector3D a jej metód nie sme schopný zmeniť ich fungovanie. Na otočenie osí by bolo potrebné komplexne analyzovať tieto metódy a matematickú logiku za nimi. Výsledok by nezodpovedal vynaloženému úsiliu.

V prípade, ak sa rozhodne zmenu spracovávať iný tím, odporúčame začať s podrobnou analýzou celej triedy Vector3D a jej nadradeným triedam.

5.1.12. Vytvorenie logiky rozlišovania vzorov

Predpokladáme, že agent sa nachádza v poli v časti 5. Keďže ihrisko je určené súradnicovou osou X a Y, môžeme v rámci testovacieho scenára umiestniť agenta na začiatok súradnicovej sústavy. Vlastnosť, ktorá by mohla byť v tomto prípade využitá je poloha agenta a rotácia resp. natočenie hráča.

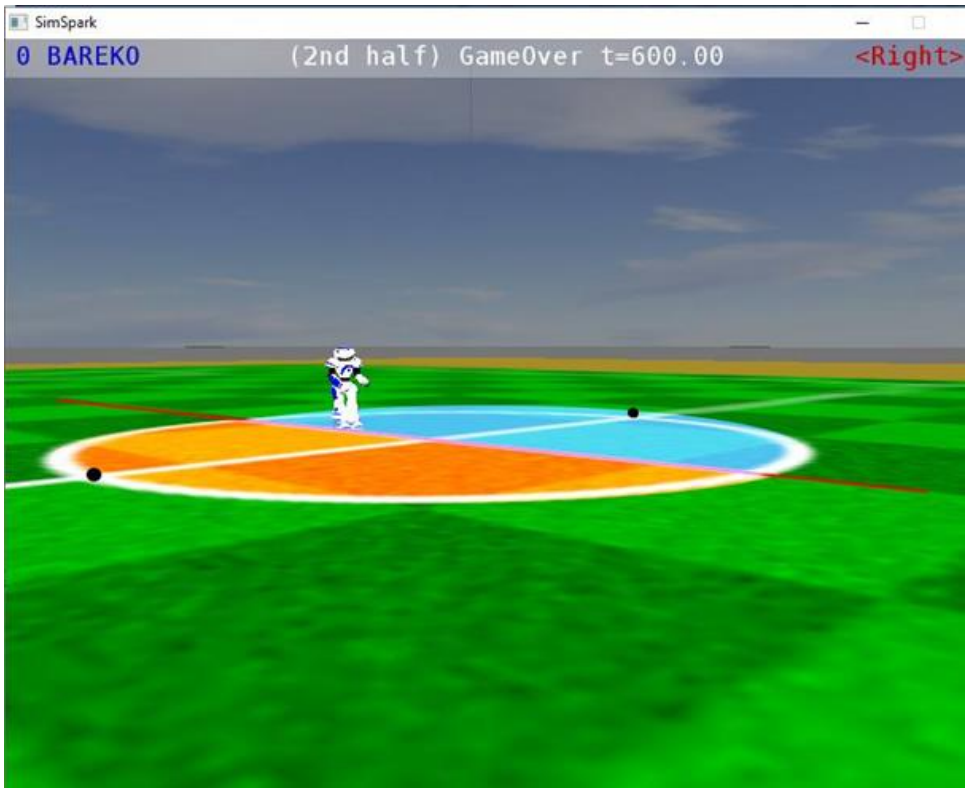
double rotationX (uhol r) - Natočenie agenta v závislosti od x-ovej osi
double rotationY (uhol phi)- Natočenie agenta v závislosti od y-ovej osi
double rotationZ (uhol theta) - Natočenie agenta v závislosti od z-ovej osi



Obrázok 38 Rozdelenie sektora na 5 základných častí

5.1.12.1. Prvý scenár

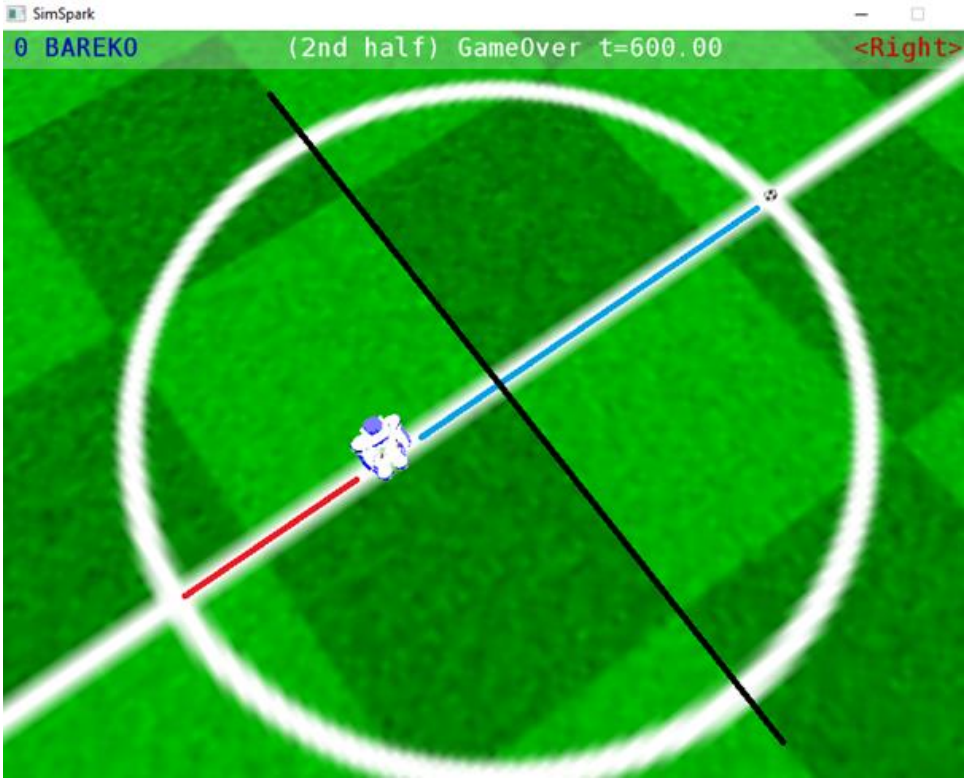
Prvým scenárom je teda hráč na začiatku súradnicovej sústavy. Na základe rotácie agenta nameranej z gyroskopu vieme definovať pri akom uhle vníma hráč iba 1 bod (horné alebo dolné "+" či "T"). Na základe nameraných uhlov a pozície hráča by sme teoreticky mali vedieť určiť o aké "+" sa jedná (viď. obrázok nižšie). Nakoľko hodnoty z gyroskopu sú však nie úplne presné, využijeme históriu pozícii z triedy *Agent.PositionHistory*, kde ukladáme údaj aj o uhle agenta. Na základe posledného nameraného uhla agenta budeme vedieť presne určiť o ktoré „+" sa jedná. Tento istý scenár rozpoznania je možné použiť aj na vzor „T".



Obrázok 39 Vizualizácia prvého scenára

5.1.12.2. Druhý scenár

Druhým možným scenárom riešenia je počítanie kladnej a zápornej vzdialenosti bodu od polohy hráča. Teda ak som agent a zo servera mi príde informácia o bode horné alebo dolné "+" resp. o jeho polohe, viem si vyrátať relatívnu vzdialenosť od mojej polohy (podobne ako tomu bolo pri určovaní horného a dolného vzoru "T" a "+"). Namerané body sa nachádzali vo vzdialenost 2.17 po y-ovej osi kladným alebo záporným smerom. Ak sa teda nachádza agent v stredovom kruhu až na 1 prípad bude vždy vedieť ako ďaleko sa nachádza od bodu na ktorý sa pozerá. Teda ak agent má kladnú relatívnu vzdialenosť od videného vzoru "+" vieme, že sa bude jednať o horné "+" a naopak ak má zápornú relatívnu vzdialenosť taks a bude jednať o dolné "+" (viď obrázok nižšie). Jediný prípad kedy nevie určiť zo vzdialenosti o ktoré "+" sa jedná je ak sa nachádza v 0 bode na X-ovej súradnici. V tomto prípade, ale môžeme znova využiť uhol natočenia hráča vzhľadom k súradnicovej sústave a zistiť či sa jedná o horný polkruh stredového kruhu alebo o ten dolný.



Obrázok 40 Vizualizácia druhého scenára

5.1.12.3. Tretí scenár

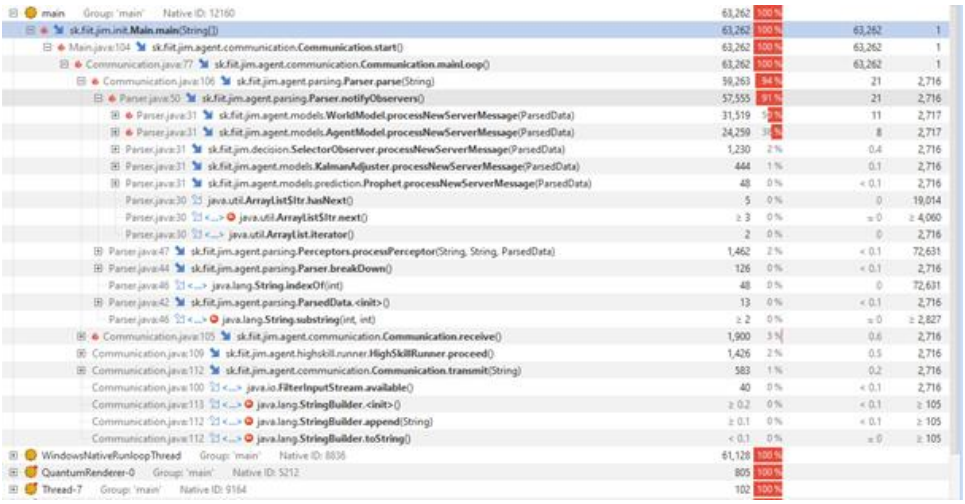
Tretím možným scenárom pre zistenie o ktoré „+“ či „T“ sa jedná je, ak vieme pozíciu hráča v piatom sektore určiť na základe už definovaných bodov, ktoré majú pridelené aj konkrétne číslo úsečky, ktorá vytvára daný vzor čiar, ako je to v prípade vzoru „T“. Ak agent bude vedieť zosnímať prvotne takýto bod a následne pokračovať ďalej, určenie o ktoré „+” sa jedná bude jasné. Ak nastane prípad, že agent neuvidí vzor “T” je stále možné odraziť sa od koncového rohového bodu, ktorý by uvidel. Vieme teda s určitosťou povedať, že agent vždy vidí aspoň jeden bod, podľa ktorého sa môže orientovať v piatom sektore a na základe neho určiť či sa jedná o dolný alebo horný vzor „+” či „T“

5.1.12.4. Agent.PositionHistory

Agent si na základe Agent.PositionHistory uchováva posledných 1000 pozícií. Ak ich prekročí, tak vymaže prvý prvok od konca. Nakoľko pamätaný rozsah pozícií je dostatočný, budeme tieto údaje využívať pri určení o ktoré „+“ či „T“ sa jedná.

5.1.13. Opis zmien po optimalizácií logov a správ do TF

Na základe analýzy pomocou nástroja Yourkit Java Profiler boli identifikované kusy kódu, ktoré sú v riešení výpočtovo najnáročnejšie. Identifikované metódy bolo potom možné upraviť, aby sa nevykonávali časti, ktoré nie sú pri niektorých úlohách nevyhnutné.



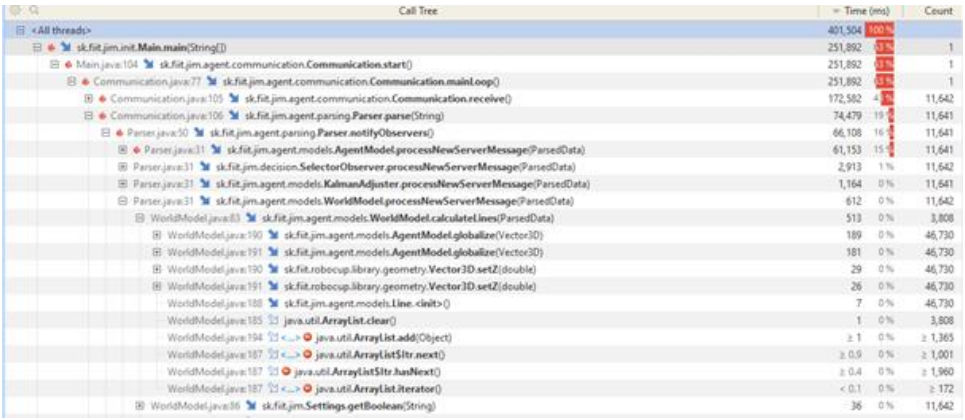
Obrázok 41 Pôvodná výpočtová zložitosť

Konkrétne ide o zmeny v metódach *WorldModel.processNewServerMessage(ParsedData)* a *AgentModel.processNewServerMessage(ParsedData),* ktorých pôvodnú zložitosť vidno na obr. č. 1. Je to 50% resp. 38%.

Po analyzovaní kódu týchto metód sa ukázalo, že vytvárajú dáta typu String aj v prípadoch, kedy sa tieto dáta ďalej nepoužijú. V prvom prípade vytvorenie správy, ktorá sa pošle do TestFrameworku a v druhom vytvorenie logového výpisu.

5.1.13.1. Optimalizácia WorldModel.processNewServerMessage(ParsedData)

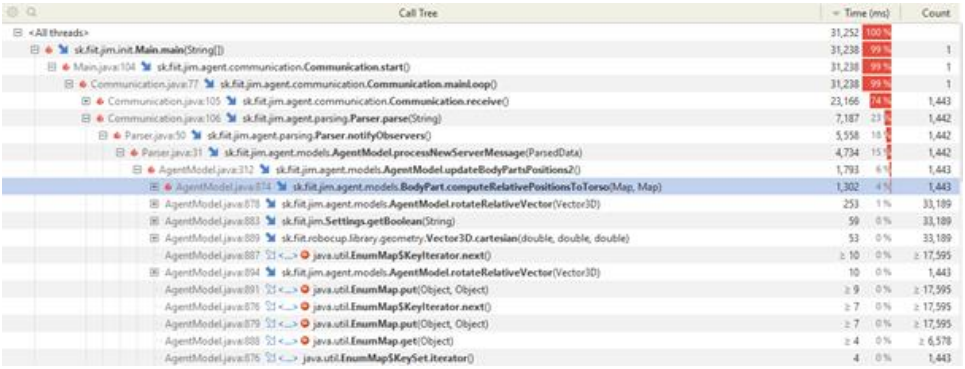
V tomto prípade sa v triede *WorldModel* zbytočne vytvára objekt typu *Message*, kde sa pre vytvorenie používa *ObjectOutputStream*, ktorý zaberá dosť času. Táto správa sa následne pošle do metódy na odoslanie do TF, kde sa ale odoslanie vykoná iba v prípade, že je TF zapnutý. Ide konkrétne o *if (Settings.getBoolean("TestFramework_monitor_enable")).* Aby sme zabránili zbytočnému vytváraniu správy, tento if sme dali rovnako aj pred vytváranie správy a následné volanie metódy na odoslanie. Funkcionalita odosielania správ do TF by teda mala byť zachovaná, ale podarilo sa nám skresať náročnosť o značnú časť, čo vyplýva z obr. č. 2.



Obrázok 42 Výpočtová zložitosť metód po vypnutí posielania správ do TF

5.1.13.2. Oprimalizácia AgentModel.processNewServerMessage(ParsedData)

V rámci tejto metódy sa volajú zmeny v chovaní a stave hráča na základe spracovaných dát zo servera. Konkrétne v metóde *updateBodyPartsPositions2(),* ktorá je počas spracovávania volaná sa nachádza vypisovanie do logu o stave hráča. Tento výpis sa zapína v GUI pomocou tlačidla AGENT_MODEL. Aj keď je tento výpis vypnutý, správa sa vytvára pomocou StringBuilder-u, čo predstavuje značné spomalenie. Aby sme sa zbytočnému vytváraniu správy vyhli, bol do triedy *Settings* pridaný atribút „logBodyPartsPosition“, ktorý určuje, či sa má logovanie použiť. V kóde je potom podmienka, kde sa kontroluje, či má daný atribút hodnotu true/false a až potom je samotný výpis do logu. Nastavenie atribútu sa deje defaultne pri spustení programu, kedy sa načíta uložená konfigurácia logovania a prepína sa pri vypnutí/zapnutí atribútu na logovanie AGENT_MODEL v GUI. Touto zmenou sme dosiahli ďalšie ušetrenie výkonu CPU.



Obrázok 43 Výpočtová zložitosť metód po vypnutí logovania

Týmito zmenami sme ušetrili čas, ktorý bol zbytočne zabratý časťami kódu, ktoré nie je potrebné vykonávať vždy. Avšak v prípade, že tieto konštrukcie sú potrebné, naďalej sa budú

v programe vykonávať. V tomto prípade ale dané metódy optimalizovať nevieme pri zachovaní pôvodnej funkcionality.

5.1.14. Optimalizácia triedy AgentModel

Cieľom úlohy je optimalizovanie triedy *sk.fiit.jim.agent.models.AgentModel* na základe analýzy Yourkitom, ktorá prvotne ukázala, že niektoré metódy v nej sú pomerne výpočtovo náročné. V tejto triede sa nachádza aktualizovanie stavu hráča, nastavenia jeho častí tela a polohy. Väčšinou ide o vektorové a podobné výpočty, ktoré sa viac optimalizovať zrejme nedajú. V triede sa ale nachádza mnoho výpisov do logov, ktoré sa volajú aj keď nie sú zapnuté. Časť týchto výpisov už bola vypnutá na základe flagu v rámci úlohy v minulom šprinte (metóda *updateBodyPartsPositions2()*), ale niektoré výpisy, ktoré tiež zaberali nepatrný čas ostali zapnuté. Sú to výpisy v metódach *updateCenterOfMass()* a *updateZeroMomentPoint()*. V rámci úlohy boli tieto výpisy takisto vypnuté, takže sa ušetrilo niekoľko percent z celkového času vykonávania.

Vypnutie logov, keď ich nie je potreba ušetrilo ~2% času. Rozdiel je viditeľný na porovnaní obr. č. 1 a obr. č. 2. Funkcionalita, keď sú logy zapnuté ostáva nezmenená.

Parser.java:31	sk.fiit.jim.agent.models.**AgentModel.processNewServerMessage**(ParsedData)	13,663	6 %
AgentModel.java:312	sk.fiit.jim.agent.models.**AgentModel.updateBodyPartsPositions2**()	4,394	2 %
AgentModel.java:313	sk.fiit.jim.agent.models.**AgentModel.updateCenterOfMass**()	2,943	1 %
AgentModel.java:315	sk.fiit.jim.agent.models.**AgentModel.updateZeroMomentPoint**()	2,597	1 %
AgentModel.java:309	sk.fiit.jim.agent.models.**AgentModel.updatePosition**(ParsedData)	1,888	1 %
AgentModel.java:304	sk.fiit.jim.agent.models.**AgentModel.adjustRotationsFor**(Vector3D)	740	0 %
AgentModel.java:306	sk.fiit.jim.agent.models.**AgentModel.updatePureBodyAcceleration**(ParsedData)	519	0 %
AgentModel.java:305	sk.fiit.jim.agent.models.**AgentModel.updateRotations**(ParsedData)	251	0 %
AgentModel.java:321	sk.fiit.jim.agent.models.**AgentModel.updateHistory**(ParsedData)	150	0 %
AgentModel.java:303	sk.fiit.jim.agent.models.**AgentModel.updateJointPositions**(ParsedData)	126	0 %
AgentModel.java:308	sk.fiit.jim.agent.models.**AgentModel.extendHistory**(ParsedData)	37	0 %
AgentModel.java:298	sk.fiit.jim.agent.models.**AgentModel.deleteHistory**(ParsedData)	7	0 %
AgentModel.java:316	sk.fiit.jim.agent.models.**AgentModel.updateSpeedX**(ParsedData)	5	0 %
AgentModel.java:314	sk.fiit.jim.agent.models.**AgentModel.updateFeetForce**(ParsedData)	1	0 %

Obrázok 44 Náročnosť metód pred optimalizáciou

Parser.java:31	sk.fiit.jim.agent.models.**AgentModel.processNewServerMessage**(ParsedData)	9,753	4 %
AgentModel.java:312	sk.fiit.jim.agent.models.**AgentModel.updateBodyPartsPositions2**()	4,773	2 %
AgentModel.java:309	sk.fiit.jim.agent.models.**AgentModel.updatePosition**(ParsedData)	2,618	1 %
AgentModel.java:304	sk.fiit.jim.agent.models.**AgentModel.adjustRotationsFor**(Vector3D)	682	0 %
AgentModel.java:306	sk.fiit.jim.agent.models.**AgentModel.updatePureBodyAcceleration**(ParsedData)	530	0 %
AgentModel.java:313	sk.fiit.jim.agent.models.**AgentModel.updateCenterOfMass**()	412	0 %
AgentModel.java:305	sk.fiit.jim.agent.models.**AgentModel.updateRotations**(ParsedData)	212	0 %
AgentModel.java:303	sk.fiit.jim.agent.models.**AgentModel.updateJointPositions**(ParsedData)	185	0 %
AgentModel.java:315	sk.fiit.jim.agent.models.**AgentModel.updateZeroMomentPoint**()	122	0 %
AgentModel.java:321	sk.fiit.jim.agent.models.**AgentModel.updateHistory**(ParsedData)	116	0 %
AgentModel.java:308	sk.fiit.jim.agent.models.**AgentModel.extendHistory**(ParsedData)	38	0 %
AgentModel.java:316	sk.fiit.jim.agent.models.**AgentModel.updateSpeedX**(ParsedData)	4	0 %
AgentModel.java:298	sk.fiit.jim.agent.models.**AgentModel.deleteHistory**(ParsedData)	3	0 %
AgentModel.java:314	sk.fiit.jim.agent.models.**AgentModel.updateFeetForce**(ParsedData)	1	0 %

Obrázok 45 Náročnosť metód po optimalizácií

5.1.15. Optimalizácia triedy LowSkills

Trieda *sk.fiit.jim.agent.moves* je podľa Yourkit nástroja výpočtovo náročnejší (*~.moves.LowSkills.step()* ma hodnotu 7%) preto je potrebné sa na to pozrieť pre prípadné zoptimalizovanie zdrojového kódu.

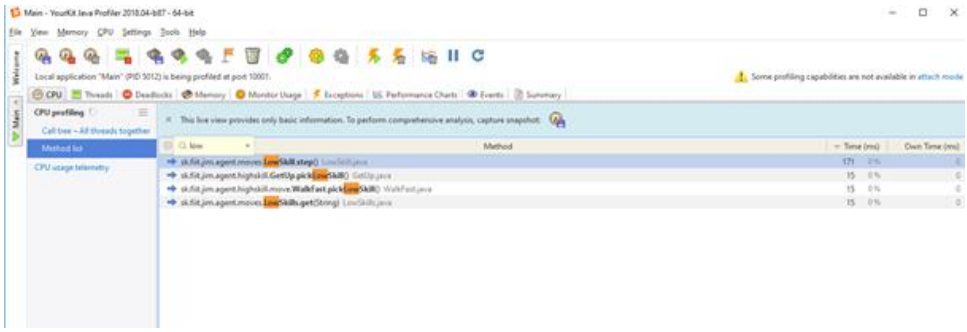
Cieľom tejto úlohy je optimalizovanie zdrojových kódov tried nachádzajúcich sa v priečinku *sk.fiit.jim.agent.moves* na základe analýzy nástroja Yourkit. Tento nástroj ukázal, že niektoré metódy v triedach sú pomerne výpočtovo náročné. V týchto triedach sa nachádza výber low skill-u hráča na základe aktuálnej vybranej taktiky a stavu. Zistili sme, že v triede *LowSkill* sa nachádzajú Log výpisy, ktoré po ošetrení podmienkou *if(Setting.getBoolean(„lowSkillsLogger“))* sa znížila výpočtová náročnosť tejto triedy na 0%. Log výpisy sa odstránili v metódach *step()* a *setNewActionPhase()*. V triede *SkipFlag* sa upravil zdrojový kód v metóde *equals()* na základe odporúčania prostredia InteliJ. Táto úprava zahŕňala zjednodušenie podmienok IF.

V metóde *NewActivePhase()* triedy *LowSkill* sa nachádza komentár:

//TODO possible stack overflow if all phases have the same skipIfFlag

Žiaľ tomuto komentáru sme neporozumeli pretože premenná *skipIfFlag* sa v kóde nenachádza a aktuálny stav zdrojové kódu nezvyšuje výpočtovú náročnosť.

Na základe analýzy sa nám podarilo znížiť výpočtovú náročnosť agenta Jim o 7%, konkrétne v triede *LowSkill* v metódach *step()* a *setNewActionPhase()*. Výsledok optimalizácie môžete vidieť na obrázku nižšie.



Obrázok 46 Yourkit po optimalizácii LowSkills

5.1.16. Opravená výminka v triede AgentPositionCalculator

Bola identifikovaná exception v triede AgentPositionCalculator keď funkcia dostala veľkosť 0. Funkcia slúži na updatnutie polohy. Vyriešilo sa to tak, že ak je veľkosť polohy 0, tak sa

neupdatne poloha, updatne sa až po ďalšej zmene. Bola implementovaná podmienka v metóde updatePosition.

5.1.17. Plánovanie trajektórie pohybov

Všeobecne sa trajektória tvorí zo zoznamu všetkých pohybov typu walk a rotate podľa ich atribútov – dĺžka kroku a uhol otočenia. Trajektória sa skladá postupne z najdlhších krokov a najväčších otočení k cieľu, pričom po každom ďalšom pridanom pohybe sa skontroluje, koľko ešte a akých veľkých pohybov je potrebné pridať na dosiahnutie cieľa.

Je treba podotknúť, že v triede je aj riešenie prekážok v chôdzi – iní hráči, ale v tomto prípade nebolo potrebné brať v úvahu trajektóriu s prekážkami.

Trajektória sa plánuje naraz pre 5 odlišných odchýliek v zmysle uhla a vzdialenosti od požadovaného cieľa. Následne sa z nich vyberie 1 najlepšia – trajektória s najkratším celkovým časom vykonávania.

V triede *sk.fiit.jim.agent.trajectory.TrajectoryPlanner* bola identifikovaná zastaralá podmienka v metóde *plan* týkajúca sa vykonania plánovanej trajektórie.

Pred úpravou sa trajektória plánovala znovu bez ohľadu na úspešnosť po prejdení ¾ vzdialenosti. Robot vďaka strate stability často spadol a celkový čas na dosiahnutie cieľa sa tak iba zväčšil.

Tento parameter sme jednoducho odstránili. Hráč takto vykoná celú plánovanú trasu a následne vypočíta prípadnú úpravu k cieľu.

Nadväznosť pohybov sa realizuje volaním metódy sequenceCheck (*sk.fiit.jim.annotation.data.MoveValidator)* v rámci metódy walk. Overí sa, či vybraný krok môže nadväzovať na už vypočítanú trajektóriu v tejto metóde. Ak áno, metóda sequenceCheck vráti hodnotu true, inak false.

Podmienky:

1. Ak agent leží == false
2. Ak pozícia na konci pohybu nesedí s pozíciou nasledujúceho pohybu a lopty
3. Ak kĺby nie sú schopné v danej pozícii vykonať nasledujúci pohyb (príliš veľké otočenie a podobne)

Ak nasledujúci pohyb trajektórie splní niektorú z podmienok 1.-3. je vyhodnotený ako nemožný / nevhodný pre pokračovanie pohybu a hľadá a iný zo zoznamu pohybov, ktorý podmienku spĺňa.

5.1.17.1. Navrhované riešenie

- Navrhujeme upraviť podmienku 1 a doplniť do nej pridanie pohybu vstávania (čiže podmienka 1 bude true).
- Podmienku 2 nechať nezmenenú (ak sa agent ocitne z nejakého dôvodu mimo plánovanú trajektóriu, je treba nájsť iný pohyb)
- Podmienku 3 upraviť na kontrolu polohy kĺbov a ich nastavenie pre daný pohyb, nie je dôvod pre hľadanie iného potenciálne menej vhodného pohybu v trajektórií

5.1.18. Implementácia funkcie JustLineSeen

V rámci tejto funkcie bola implementovaná nasledujúca logika (prvá verzia):

1. hľadanie dvojíc čiar, ktoré sa pretínajú
2. ak sa nájde zhoda, hľadanie končí a čiary sa uložia
3. získanie súradníc priesečníka
4. zistenie, v ktorej časti ihriska sa hráč nachádza
5. porovnávanie so vzormi čiar, na ktoré môže v rámci funkcie naraziť (ostatné prípady budú obslúžené ostatnými funkciami)
6. zavolanie funkcií na zisťovanie vzorov (napr. isT())
7. podľa toho, v ktorej časti ihriska sa nachádza hráč zistenie, o ktorý bod ide
8. ak sa hráč nachádza v hornej/dolnej časti a relatívna súradnica Y je menšia ako Y vzdialenosť daného bodu od stredu ihriska, bod sa nachádza v rovnakej polovici
9. ak je relatívna súradnica Y väčšia ako Y vzdialenosť daného bodu od stredu ihriska, bod sa nachádza v opačnej polovici
10. logika keď sa hráč nachádza presne medzi dvoma bodmi (podľa osi Y) ešte implementovaná nie je
11. vytvorenie namapovaného objektu bodu s relatívnymi a absolútnymi súradnicami a pridanie do výsledného poľa
12. Výsledok funkcie je takisto vždy zaznamenaný do logu, aby bolo možné funkcionalitu v budúcnosti testovať.

Popri tejto funkcií boli aj vytvorené funkcie LinePatternRecognition.isHorizontalT() a LinePatternRecognition.isVerticalT() na zistenie, či ide o T na krajoch v strede ihriska alebo o T tvoriace časť bránkoviska.

Na otestovanie bol vytvorený unit test AgentPositionCalculatorTest, pomocou ktorého bola implementácia prvotne otestovaná. Taktiež môže byť použitý pri ďalšom testovaní s použitím iných kombinácií čiar resp. vzorov.

Implementovaná funkcia je schopná rozoznať medzi čiarami tie, ktoré sa pretínajú a zistiť, aký vzor tvoria. Popri použití ostatných funkcií na zisťovanie polohy (vidí len jednu čiaru, vidí kontrolný bod) napomáha presnejšiemu zisťovaniu polohy, keď ostatné funkcie nemožno použiť. Testovanie môže prebiehať buď pomocou unit testu alebo popri reálnom behu hry a kontrolovaniu logov, kde sa výsledok funkcie zapisuje. Funkcia je zatiaľ v prvej verzii a na základe výsledkov testu a ďalších analýz je možné ju v budúcnosti optimalizovať a vylepšiť jej presnosť (funkcie zisťovania vzorov zatiaľ neuvažujú skreslenie). Môže byť aj pridané mapovanie videných koncových bodov čiar na základe absolútnych súradníc identifikovaných bodov, po vzore funkcie AgentPositionCalculator.oneFixedObjectSeen().

5.1.19. Implementácia funkcie isIntersection

Cieľom úlohy je implementovať funkciu isIntersection, ktorá zisťuje, či majú dve úsečky spoločný priesečník v tvare + alebo T. Táto funkcia bola predošlým tímom identifikovaná ako potrebná a doposiaľ neimplementovaná. Na jej základe môže hráč lepšie určiť svoju polohu – podľa relatívnych súradníc, ktoré dostane od servera vypočíta, o aký tvar úsečiek ide a podľa vzdialenosti zistí, kde sa nachádza.

Pre naplnenie úlohy boli implementované 2 metódy v triede LinePatternRecognition, metóda *isIntersection() a getSegmentIntersect()*. Správnosť týchto metód je otestovaná unit testom v tiede LinePatternRecognitionTest.

5.1.19.1. Metóda isIntersection

```
public static boolean isIntersection(ParsedLineWithFlags, ParsedLineWithFlags)
```

V tejto metóde sa najskôr zistí, či dané úsečky, ktoré prídu ako parameter (v tvare x1, y1, x2, y2) majú priesečník a to pomocou vstavanej metódy *intersectsLine()* triedy Line2D (java.awt trieda, nie trieda z projektu). Ak úsečky majú priesečník, pomocou metódy *getSegmentIntersect()* sa vypočítajú jeho súradnice a následne sa zisťuje, či nejde o úsečky

tvoriace roh. To sa vyhodnocuje na základe porovnania súradníc priesečníka a jednotlivých bodov obidvoch úsečiek. Metóda má návratovú hodnotu typu boolean, ktorá nadobúda hodnoty podľa toho, či majú úsečky priesečník a či sa jedná o roh.

5.1.19.2. Metóda getSegmentIntersect

```
public static Point2D getSegmentIntersect(java.awt.geom.Line2D,
                                          java.awt.geom.Line2D)
```

Pomocou tejto metódy sa vypočítajú súradnice priesečníka dvoch úsečiek metódou na základe vektorového súčinu. Metóda vracia priesečník reprezentovaný triedou Point2D, v ktorej sú jeho súradnice x a y. Ak dané úsečky priesečník nemajú, metóda vráti null.

Základné testovanie vypracovaných metód sa nachádza v triede LinePatternRecognitionTest, kde sa už nachádzajú testy ostatných existujúcich metód modifikovanej triedy. Pre účely testovania nových metód sú v teste vytvorené nové objekty čiar, ktoré sa buď nepretínajú, alebo tvoria priesečník v tvare +, T alebo roh. Správnosť výsledku metódy je potom skontrolovaná porovnaním s očakávanou hodnotou. Príklad volania:

assertEquals(false, isIntersection(lines.get(6), lines.get(7)));

Pre potreby testovania bola takisto pozmenená trieda Vector3D, v ktorej sa nenachádzali settery pre súradnice, napr. *setX()*.

Pomocou implementovanej metódy *isIntersection()* je možné zistiť, či dve úsečky majú priesečník tvaru, ktorý je pre vyhodnotenie polohy hráča relevantný a teda je možné ju použiť na zlepšenie orientácie hráča na ihrisku. Pomocná metóda *getSegmentIntersect()*, ktorá bola implementovaná pre potreby úlohy sa dá použiť aj v iných implementáciách pre získanie priesečníka dvoch úsečiek (metóda na získanie priesečníka dvoch priamok sa už v kóde nachádza).

5.1.20. Implementácia funckie isGoalBox

Cieľom úlohy je pridanie funkcie isGoalBox do triedy *sk.fiit.jim.agent.models.LinePatternRecognition.* Funkcia zisťuje, či dané dve čiary sa pretínajú vo vzore „goalBox“.

Táto funkcia zistuje, či sú tri vstupujúce čiary bránkovisko. Keďže nevie, ktorá čiara je ktorá, je treba zistiť, medzi ktorými dvoma čiarami sa nachádza vzor roh (funkciou isCorner). Tieto dve čiary sú čiary bránkoviska - šestnástky. Následne sa pre obe tieto čiary skontroluje, či s

treťou čiarou netvoria vzor T (funkciou isT). Ak sa zistí že sa jedná o bránkovisko, tak sa vráti usporiadaná trojica, kde na prvej pozícii (indexe 0) je číslo zadnej čiary, na druhej pozícii (indexe 1) je je číslo hrany tvoriacou T so zadnou čiarou, a na tretej pozícii (indexe 2) je číslo poslednej, prednej čiary.

Čiara na indexe 1 má priradené záporné číslo ak spoločný bod čiar bránkoviska je druhým bodom bočnej čiary, a kladné číslo ak sa jedná o prvý bod. Podobne pre čiaru na indexe 2 platí, že ak je tento spoločný bod druhým bodom, vrátená hodnota bude záporná, a ak prvým bodom, tak hodnota bude kladná.

Funkcia teda môže vrátiť 6 rôznych permutácií poľa [1,2,3], pričom druhá a tretia hodnota môžu byť záporné - celkovo 24 rôznych hodnôt, plus null v prípade, že sa nejedná o bránkovisko.

5.1.21. Implementácie funkcie isT

Cieľom úlohy je pridanie funkcie isT do triedy *sk.fiit.jim.agent.models.LinePatternRecognition.* Funkcia zisťuje, či dané dve čiary sa pretínajú vo vzore „T“.

Implementovali sme funkciu isT, do ktorej vstupujú dve čiary (obe sú definované dvoma koncovými bodmi).). Z týchto dvoch bodov sa vypočítajú rovnice (priamok) týchto čiar v tvare y = ax + b.

Pre tieto rovnice priamok sa vypočíta ich priesečník. Tento priesečník by mal byť bodom, v ktorom sa vzor T nachádza.

Následne sa pre všetky 4 body, ktoré vstúpili do funkcie, zistí, či sú totozné s týmto novým bodom. Na túto kontrolu slúži funkcia isSamePoint, ktorá by mala pracovať aj s potenciálnou chybou.

V prípade, že sa zistí, že sa jedná o bod ktorý leží na priesešníku čiar sa ešte skontroluje, či ani jeden bod z druhej čiary neleží na tomto bode (jednalo by sa o vzor corner nie T).

Funkcia *isT()* vracia hodnoty:

0 – nie je vzor T

1 – prvý bod prvej čiary leží na T

2 – druhý bod prvej čiary leží na T

3 – prvý bod druhej čiary leží na T

4 – druhý bod druhej čiary leží na T

5.1.22. Implementácia funkcie isPlus

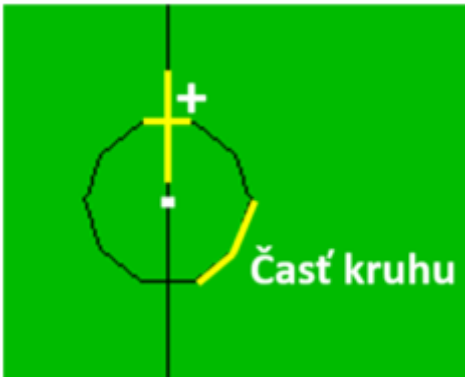
Cieľom úlohy je pridanie funkcie isPlus do triedy *sk.fiit.jim.agent.models.LinePatternRecognition.* Funkcia zisťuje, či dané dve čiary sa pretínajú vo vzore „+“.

Implementovali sme funkciu *isPlus*(), ktorá zisťuje pomocou metódy *isIntersection()*, či dané dve čiary sa pretínajú a zároveň netvoria roh. Zároveň sa zisťuje, že ak sa pretínajú dané dve čiary tak nech netvoria vzor „T“, toto sa ošetruje volaním funkcie *isT()*, ktorá vracia hodnotu nula ak dve čiary netvoria vzor „T“. Ďalej sa zisťuje či existuje bod priesečníka pomocou metódy *getSegmentIntersect()*medzi dvomi čiarami. Bod musí existovať. Ak sa všetky predchádzajúce podmienky splnia tak vzor plus existuje. Stredová čiara je stále prvá čiara, ktorá sa posiela zo servera. Do funkcie *isPlus()*sa majú posielať iba čiary 1,14,19 pretože ináč sa nedá zistiť, že ktorá je stredová čiara. Inštancia čiary nemá informáciu v sebe o type čiary alebo poradí. Toto sa má ošetriť pred volaním samotnej funkcie pri spracovaní odpovede zo servera.

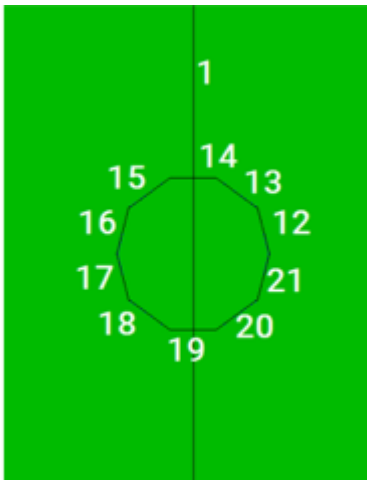
Funkcia *isPlus()* vracia hodnoty:

0 – nie je vzor plus

1 – je vzor plus a prvá čiara je stredová čiara



Obrázok 48 Vzor plus

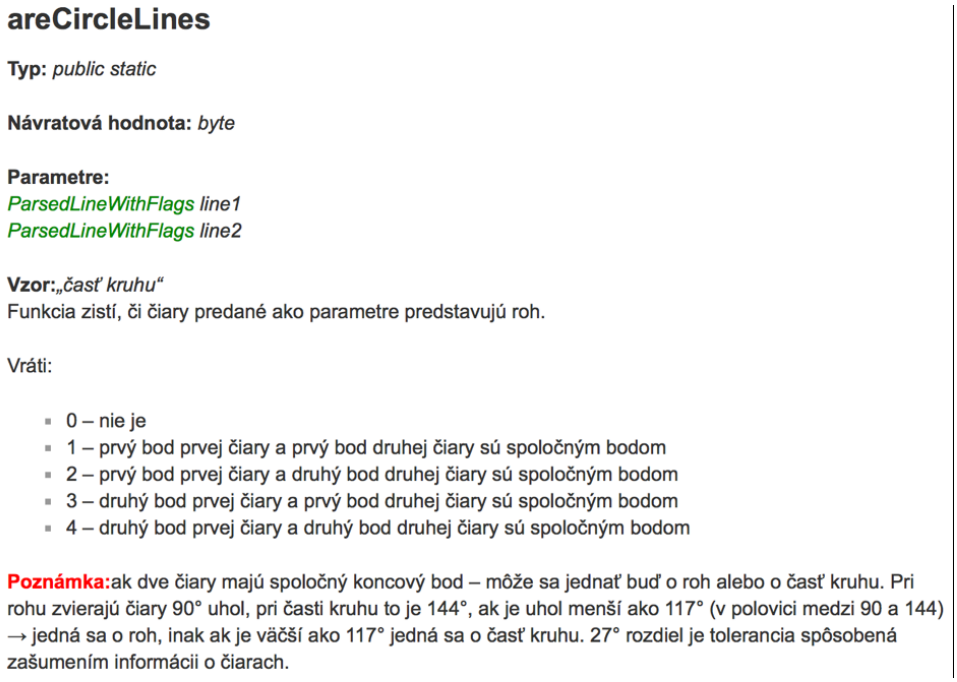


Obrázok 47 Poradie čiar

5.1.23. Funkcia pre zistenie či sú dve čiary súčasťou stredového kruhu

Pre túto úlohu je potrebné zapracovať metódu či sú dve čiary súčasťou stredového kruhu do triedy *JustLineSeen*. Taktiež je potrebná úprava komentárov pre jednotlivé, už existujúce metódy v danej triede.

Danú metódu na výpočet či sú dve čiary súčasťou stredového kruhu nie je potrebné zapracovať lebo už existuje vo forme metódy *areCircleLines()*, viď. obrázok nižšie.



areCircleLines

Typ: *public static*

Návratová hodnota: *byte*

Parametre:
ParsedLineWithFlags line1
ParsedLineWithFlags line2

Vzor:*„časť kruhu“*
Funkcia zistí, či čiary predané ako parametre predstavujú roh.

Vráti:

- 0 – nie je
- 1 – prvý bod prvej čiary a prvý bod druhej čiary sú spoločným bodom
- 2 – prvý bod prvej čiary a druhý bod druhej čiary sú spoločným bodom
- 3 – druhý bod prvej čiary a prvý bod druhej čiary sú spoločným bodom
- 4 – druhý bod prvej čiary a druhý bod druhej čiary sú spoločným bodom

Poznámka:ak dve čiary majú spoločný koncový bod – môže sa jednať buď o roh alebo o časť kruhu. Pri rohu zvierajú čiary 90° uhol, pri časti kruhu to je 144°, ak je uhol menší ako 117° (v polovici medzi 90 a 144) → jedná sa o roh, inak ak je väčší ako 117° jedná sa o časť kruhu. 27° rozdiel je tolerancia spôsobená zašumením informácii o čiarach.

Obrázok 49 Metóda areCircleLines()

Ďalej sme upravili komentáre pre jednotlivé metódy. Komentáre sedia s návratovými hodnotami jednotlivých metód. Trieda JustLineSeenobsahuje dve metódy na zistenie znaku „T“: isT(), isPatternT(). Ich funkcionalita je rovnaká no odporúčam využívať naďalej metódu isT(). Rozdiel je vo forme vyhodnocovania znaku „T“ a v metóde isPatternT()je o jednu návratovú hodnotu viac (return 5), ktorá hovorí o tom, že znak je pravdepodobne „+“. Po úprave komentárov v triede LinePatternRecognitionsme zistili, že všetky metódy na zapracovanie (označené TODO) sú vyriešené.

5.1.24. Vytvorenie taktiky pre testovanie rozlišovania vzorov

Cieľom úlohy je vytvorenie taktiky, ktorá sa použije na účely testovania predošle implementovaných funkcií na zisťovanie polohy hráča na základe rozpoznávania tvarov čiar, konkrétne metódu ***AgentPositionCalculator.justLinesSeen()***. Toto testovanie by malo

prebiehať za normálneho behu hry, kde výsledky rozpoznávania tvarov budú vypísané do logov.

Hlavným ťažiskom je vytvorenie triedy ***PositionTestTactic***, ktorá obsahuje otočenie hráča o 90° v smere v závislosti od toho, na ktorej polovici sa nachádza. Keď je hráč v hornej polovici ihriska, otočí sa doľava, aby videl iba požadovaný bod. Po otočení zostane stáť bez pohybu. Takisto aby bolo logovanie rozpoznávania prehľadnejšie, bol vytvorený nový level logovania POINTS, ktorý sa dá zapnúť cez GUI ako ostatné levely.

Taktika sa dá použiť v spojení nastavovania štartovacej pozície hráča (možné iba na začiatku hry a v polčase) v triede ***MatchStarterTactic***, kde sa dajú nastaviť v metóde ***runBeam()*** súradnice hráča (s id 1). To sa dá použiť na postavenie hráča na ľubovoľnú pozíciu v rámci polovice ihriska jeho tímu. Keď sa hráč postaví do tesnej blízkosti bodu a otočí sa, mal by vidieť iba tento bod a v tom prípade sa pustí výpočet pozície na základe rozpoznávania tvarov čiar.

Pomocou implementovanej taktiky by malo byť jednoduchšie otestovať rozpoznávacie techniky vytvorené v predošlých šprintoch. Možno budú potrebné ešte ďalšie zmeny pre identifikovanie ostatných bodov, takže taktika sa počas testovania bude pravdepodobne meniť.

5.1.25. Poslanie príkazu exit pre agentov nespustených v testovacom frameworku

Na základe komentára (viď. nižšie) v triede sk.fiit.testframework.communication.agent.AgentManager.java sme mali implementovať posielanie ukončovacieho príkazu pre agentov, ktorý neboli spustený testovacím frameworkom.

Removes an agent that was started by the test framework and terminates its process.

Does nothing for agents that were not launched by the test framework.

@param agent the agent to remove

TODO send an exit command to agents not launched by the test framework

Na základe konzultácií na stretnutí s vedúcim projektu sme sa dohodli o zamietnutí/nevypracovaní tejto úlohy, pretože dôležitosť riešenia tejto úlohy nie je podstatná a nemá dopad resp. prínos pre iné úlohy, ktoré riešime. Úloha bola nakoniec zatvorená a komentáre v kóde triedy AgentManager boli doplnené.

5.1.26. Úprava logiky spracovania taktík a stratégii

Analýza súčasného riešenia bola časovo náročná, nakoľko sa tieto štruktúry veľmi často používali pri rozhodovaní o stratégiách a taktikách. Nakoľko list obsahoval iba unikátne hodnoty, rozhodli sme sa pre použitie údajovej štruktúry Set. Tá je uspôsobená na uchovávanie práve unikátnych hodnôt.

Ďalším krokom analýzy bolo sledovanie obsahu jednotlivých štruktúr. Očakávali sme, že menší list je podmnožinou väčšieho, čo sa ukázalo ako nepravda – z toho dôvodu sme museli zavrhnúť pôvodné riešenie s porovnávaním veľkostí. Algoritmus sme však upravil tak, aby zhodné hodnoty vyhľadával z menšej štruktúry kvôli vyššej rýchlosti.

Metódy s názvom getSuitability by mali mať rovnakú funkcionalitu, no stretli sme sa s 3 rôznymi implementáciami, ktoré dosahovali iné výsledky. Tieto implementácie sme upravili tak, aby ich výpočet bol ekvivalentný pre rovnaké vstupné dáta.

Nové riešenie opravilo chyby predchádzajúcich výpočtov a počas testovania sme sa nestretli so žiadnou komplikáciou. Rýchlosť rozhodovania pre stratégiu by mala byť omnoho vyššia. Celková úprava štruktúr ovplyvnila 52 tried.

5.1.27. Nahradenie lowskillu „hlavaruky“

Prvou podstatnou časťou bolo zistiť prečo sa vôbec Jim do stavu kedy začne tento pohyb vykonávať dostane a aký lowskill by bolo najlepšie v danej situácii využiť. S dlhodobého skúmania správania Jima a dôvodu prečo sa začne vykonávať tento pohyb sme zistili, že najvhodnejšie bude namiesto priameho LowSkillu vykonávať HighSkill Localize. Tento skill Jimovi umožní obnoviť svoj prehľad o lokácii a pomôže mu s návratom do hry. Na toto sme museli však upraviť/pridať viaceré xml súbory a opraviť 2 triedy.

Ovplyvnené xml súbory sú head_down.xml, head_left.xml, head_right.xml a head_left_and_right.xml.

Ovplyvnené boli tieto triedy:

Walk.java – táto trieda obsahovala rozhodovací strom v ktorom sa na konci dostal Jim k pokynu vykonávania spartakiády. Toto bolo napravené pridaním HighSkillu “Localize” do zoznamu vykonávaných HighSkillov na začiatok a následné vyvolanie LowSkillu “rollback”.

Localize.java – tento HighSkill bol vo vysokej miere upravený – nastalo taktiež pridanie možnosti pozrieť sa dole. Taktiež bola upravená rozhodovacia štruktúra. Pridaním rozhodnutia

o pozrení dole bolo pre efektívnosť vhodné pridať metódu, ktorá resetuje históriu vykonania pohybov hlavou pri zmene polohy, respektíve opätovné vykonanie pohybu hlavou v smere pohybu a dole.

5.1.28. Upravenie fitness hodnoty pre silu kopu

Compute relative fitness to strength of kick (annotations) je TODO, ktoré sa nachádza v triede TacticalInfo v opise metódy getPassFitness. K tomuto TODO nebol žiadny ďalší komentár.

V metóde getPassFitness sa počíta fitness funkcia, ktorá by mala vyjadrovať s akou „pravdepodobnosťou“ bude prihrávka na určité miesto úspešná – nebude zachytená protihráčom. Táto fitness funkcia sa počíta na základe uhlu medzi protihráčom, miestom z ktorého sa prihráva a miestom, na ktoré sa prihráva. Hodnoty tejto fitness funkcie sú medzi 0 a 1.

Výsledná fitness funkcia je teda vypočítana na základe najmenšieho uhlu medzi smerom kopu a protihráčom.

Keďže sa hodnota počíta len z uhlov hráča,všetky typy prihrávok majú rovnakú fitness. Avšak pravdepodobnosť, že protihráč zachytí prihrávku klesá so silou kopu.

V triede kick sa nachádzajú nasledovné typy kopu:

KICK_FAST

-speedKoef = 2.0
-distanceKoef = 1.0

KICK_NORMAL
-speedKoef = 1.0
-distanceKoef = 1.0

KICK_SLOW
-speedKoef = 1.0
-distanceKoef = 0.5

KICK_STRONG
-speedKoef = 1.0

-distanceKoef = 1.5

KICK_UNKNOWN

-speedKoef = 1.5

-distanceKoef = 1.5

Každý typ kopu má tri atribúty – speedKoef, distanceKoef a succesKoef, avšak succesKoef je pre všetky typy nastavený na rovnakú hodnotu – 1.0. SpeedKoef a distanceKoef vyjadrujú rýchlosť kopu (touto rýchlosťou sa myslí rýchlosť vykonania pohybu) a vzialenosť na akú sa kope. Inicializované su na normal kick (1.0 a 1.0) a pri rôznych typoch kopov sa menia. Najzaujímavejší atribút pre túto fitness hodnotu je teda distanceKoef, s tým, že rýchlosť vykonania kopu tiež môže ovplyvniť úspešnosť nahrávky.

Na základe použitého typu kopu na nahrávku by bolo treba upraviť fitness hodnotu. Najjednoduchším riešením by bolo priradiť každému typu kopu hodnotu, ktorou by sa fitness prenásobila (napr hodnota 1 pre kick_strong a hodnota 0.5 pre kick_slow, pre ostatné typy niekde medzi). Tieto hodnoty by následne mohli byť upravené po testovaní. Ďalšou otázkou nad ktorou by sa bolo dobré zamyslieť je, či by nebolo vhodné počítať aj so vzialenosťou protihráča, nie len uhlom.

Úloha zatiaľ nie je naprogramovaná, keďže náš agent (Jim) ešte nehrá proti protihráčom, takže by sme úlohu nemali ako otestovať. Vypracovanie úlohy slúži ako podklad pre prácu tímu, ktorý bude úlohu programovať.

5.1.29. Vymazanie nepoužívaných súborov a konfigurácii IDE

Počas prehľadávania sme z Moves odstránili nepoužívané súbory, ktoré nemali žiadnu výpovednú hodnotu alebo neočakávame ich opätovné vyžitie. Tieto súbory sú:

- drepyII.xml
- filipskuska.xml
- hlavaaruky.xml
- nahlavu.xml
- rollback2.xml
- ruky.xml
- shot_left.xml
- sit_down.xml

- step_circ_left.xml
- step_circ_left_small.xml
- step_circ_right.xml
- step_circ_right_small.xml
- test_turn_left.xml
- test_turn_right.xml
- uvidime.xml
- walk_fine.xml
- walk_fine_fast.xml
- walk_turn_left.xml
- walk_turn_right.xml

Ďalšie súbory sa nachádzali buď v priečinkoch označených ako „OLD“. Ďalšie väčšia množina súborov, ktoré boli odstránené boli súbory, ktoré si predefinujú programovacie editory – tieto súbory sú po zavedení Mavenu nepodstatné. Odstránených bolo cez 400 súborov a boli vykonané menšie modifikácie 5 tried.

5.1.30. Vymazanie nepoužitých importov

Cieľom úlohy bolo vymazať nepoužívané a nepotrebné importy. Bol použitý nástroj v IntelliJ Idea IDE na inšpekciu kódu. Konkrétne Code -> Optimize code. Postupne potom boli skontrolované, vymazané alebo upravené importy v triedach.

Zoznam modifikovaných tried:

Jim/src/sk/fiit/jim/Settings.java
Jim/src/sk/fiit/jim/agent/AgentInfo.java
Jim/src/sk/fiit/jim/agent/communication/Communication.java
Jim/src/sk/fiit/jim/agent/communication/testframework/Message.java
Jim/src/sk/fiit/jim/agent/communication/testframework/TestFrameworkCommunication.java
Jim/src/sk/fiit/jim/agent/highskill/Bending.java
Jim/src/sk/fiit/jim/agent/highskill/KickTestHighSkill.java
Jim/src/sk/fiit/jim/agent/highskill/kick/Kick.java
Jim/src/sk/fiit/jim/agent/highskill/move/MovementSkills.java
Jim/src/sk/fiit/jim/agent/highskill/move/Walk.java
Jim/src/sk/fiit/jim/agent/highskill/move/WalkFastZMP.java
Jim/src/sk/fiit/jim/agent/highskill/move/WalkFastZMPOld.java
Jim/src/sk/fiit/jim/agent/highskill/runner/HighSkillPlanner.java
Jim/src/sk/fiit/jim/agent/kalman/KalmanExample.java
A ďalšie ...

Po vymazaní importov knižníc by sa program nemal stať akýmkoľvek spôsobom nefunkčný alebo poškodený. Overenie v podobe krátkeho otestovania funkčnosť potvrdilo.

5.1.31. Vymazanie nepoužitých metód časť 1

Cieľom úlohy je vymazať nepoužívané a nepotrebné metódy aby sa sprehľadnil kód. Takto bude aj pre budúce tímy jednoduchšie sa v kóde orientovať a nebudú musieť zbytočne strácať čas nad metódami, ktoré sa nevyužívajú.

Bol použitý nástroj v Idea IDE na inšpekciu kódu. Konkrétne Analyze -> Inspect Code -> Inspection Profile (…) -> Declaration redundancy -> Unused declaration, označenie ako chyba. Postupne potom boli skontrolované a vymazané metódy z tried začínajúcim na písmeno A až M. Väčšinou išlo o staré triedy, ktoré sa prestali používať (a nie je k nim dostupné žiadne info) pravdepodobne z dôvodu nepotrebnosti a boli nahradené novými. Taktiež bola vymazaná jedna trieda, ktorá sa nepoužívala.

Zoznam modifikovaných tried:

Jim\src\sk\fiit\jim\agent\highskill\Bending.java
Jim\src\sk\fiit\jim\agent\highskill\HeadDownSkill.java

Jim\src\sk\fiit\jim\agent\models\AgentModel.java

Jim\src\sk\fiit\jim\agent\models\BodyPart.java

Jim\src\sk\fiit\jim\agent\skills\HighSkill.java

Jim\src\sk\fiit\jim\agent\skills\HighSkillUtils.java

Jim\src\sk\fiit\jim\annotation\data\MoveValidator.java

Jim\src\sk\fiit\jim\log\JimLocalCsvFileCreator.java

Jim\src\sk\fiit\jim\log\JimLocalFileCreator.java

RoboCupLibrary\src\sk\fiit\robocup\library\geometry\Angles.java

TestFramework\src\sk\fiit\testframework\annotator\serialization\MoveValidator.java

TestFramework\src\sk\fiit\testframework\beta\ui\ComparingTab.java

TestFramework\src\sk\fiit\testframework\communication\agent\AgentManager.java

TestFramework\src\sk\fiit\testframework\init\Implementation.java

TestFramework\src\sk\fiit\testframework\monitor\AgentMonitor.java

TestFramework\src\sk\fiit\testframework\monitor\AgentMonitorMessage.java

TestFramework\test\sk\fiit\testframework\MessageParserTest.java

Vymazaná trieda:

TestFramework\src\sk\fiit\testframework\incomplete\GoalieTestCase.java

Niektoré nájdené metódy vymazané neboli. Išlo hlavne o konštruktory GUI tried, ktoré inšpekcia taktiež označila za nepoužívané, avšak ich vymazanie by znefunkčnilo GUI.

Po vymazaní metód by program nemal byť akýmkoľvek spôsobom nefunkčný alebo poškodený. Overenie v podobe krátkeho otestovania funkčnosť potvrdilo. Kód by po dokončení ostatných úloh na vymazanie zbytočnosti mal byť čistejší a ľahšie sa v ňom orientovalo.

5.1.32. Vymazanie nepoužitých metód časť 2

Cieľom úlohy je vymazať nepoužívané a nepotrebné metódy aby sa sprehľadnil kód. Takto bude aj pre budúce tímy jednoduchšie sa v kóde orientovať a nebudú musieť zbytočne strácať čas nad metódami, ktoré sa nevyužívajú. Bol použitý nástroj v Idea IDE na inšpekciu kódu. Konkrétne Analyze -> Inspect Code -> Inspection Profile (…) -> Declaration redundancy -> Unused declaration, označenie ako chyba. Postupne potom boli skontrolované a vymazané metódy z tried začínajúcim na písmeno N až Z. Väčšinou išlo o staré triedy, ktoré sa prestali používať (a nie je k nim dostupné žiadne info) pravdepodobne z dôvodu nepotrebnosti a boli nahradené novými.

Zoznam modifikovaných tried:

Jim\src\sk\fiit\jim\agent\models\TacticalInfo.java

Jim\src\sk\fiit\jim\agent\models\WorldModel.java

Jim\src\sk\fiit\jim\agent\parsing\Perceptors.java

RoboCupLibrary\src\sk\fiit\robocup\library\geometry\Point2D.java

TestFramework\src\sk\fiit\testframework\annotator\serialization\XMLcreator.java

TestFramework\src\sk\fiit\testframework\communication\robocupserver\RobocupServer.java

TestFramework\src\sk\fiit\testframework\trainer\testsuite\TestCaseHelper.java

Vymazané triedy:

Jim\src\sk\fiit\jim\decision\situation\NoOneHasBall.java

Jim\src\sk\fiit\jim\annotation\data\XMLCreator.java

Jim\src\sk\fiit\jim\agent\parsing\SeePerceptor.java

TestFramework\src\sk\fiit\testframework\annotator\serialization\XMLparser.java

Niektoré nájdené metódy vymazané neboli. Išlo hlavne o konštruktory GUI tried, ktoré inšpekcia taktiež označila za nepoužívané, avšak ich vymazanie by znefunkčnilo GUI. Niektoré nepoužívané metódy alebo triedy obsahovali komentáre, aby neboli vymazané. Po vymazaní metód by program nemal byť akýmkoľvek spôsobom nefunkčný alebo poškodený. Overenie v podobe krátkeho otestovania funkčnosť potvrdilo. Kód by po dokončení ostatných úloh na vymazanie zbytočnosti mal byť čistejší a ľahšie sa v ňom orientovalo.

5.1.33. Vymazanie nepoužívaných súborov v konfigurácii IDE

Na gite sa nachádza veľké množstvo súborov, ktoré sa nevyužívajú a nie sú pre projekt potrebné. Počas prehľadávania sa z Moves odstránili nepoužívané súbory, ktoré nemali žiadnu výpovednú hodnotu alebo neočakávame ich opätovné vyžitie. Tieto súbory sú:

- drepyII.xml
- filipskuska.xml
- hlavaaruky.xml
- nahlavu.xml
- rollback2.xml
- ruky.xml
- shot_left.xml
- sit_down.xml
- step_circ_left.xml
- step_circ_left_small.xml
- step_circ_right.xml
- step_circ_right_small.xml
- test_turn_left.xml
- test_turn_right.xml

- uvidime.xml
- walk_fine.xml
- walk_fine_fast.xml
- walk_turn_left.xml
- walk_turn_right.xml

Ďalšie súbory sa nachádzali v priečinkoch označených ako „OLD“. Ďalšia väčšia množina súborov, ktoré boli odstránené boli súbory, ktoré si predefinujú programovacie editory – tieto súbory sú po zavedení Mavenu nepodstatné. Bolo odstránených cez 400 súborov, menšie modifikácie 5 tried.

5.1.34. Implement prediction correctly

Ide to vypracovanie jedného z TODO, ktoré boli pridané niektorým z predošlých tímov. Úloha sa viaže k triede sk.fiit.jim.agent.models.prediction.Prophet, kde sa vypočítava budúca poloha lopty a hráčov. Trieda bola vytvorená ako experiment a tím si nebol istý, či funguje správne. Po otestovaní výpočtov bolo zistené, že kvôli nepresnostiam v ostatných výpočtoch polohy, nie sú ani tieto úplne presné. Vypočítané hodnoty sa používajú pri odhadovaní stratégie v triede sk.fiit.jim.agent.models.TacticalInfo. Tieto odhady však nikde nie sú volané, čiže reálne sa táto funkcionalita momentálne nepoužíva. V triede neboli vykonané žiadne zmeny ale zatiaľ ju nevymažeme, pretože by mohla byť použitá v budúcnosti.

5.1.35. Úloha change conditions in method sequenceCheck

Analýzou fungovania plánovania trajektórie v triede TrajectoryPlanner a MoveValidator sme zistili, že metóda sequenceCheck nie je úplná a mala v sebe priestor na zmeny – úpravu podmienok pri zlej nadväznosti pohybov. Navrhujeme upraviť podmienku 1. a doplniť do nej pridanie pohybu vstávania (čiže podmienka 1. bude true). Podmienku 2. nechať nezmenenú (ak sa agent ocitne z nejakého dôvodu mimo plánovanú trajektóriu, je treba nájsť iný pohyb). Podmienku 3. upraviť na kontrolu polohy kĺbov a ich nastavenie pre daný pohyb, nie je dôvod pre hľadanie iného potenciálne menej vhodného pohybu v trajektórií

Podmienka 1. bola pridaná do triedy sk.fiit.jim.annotation.data.MoveValidator.java.
V kóde boli pridané riadky v metóde sequenceCheck:
HighSkillPlanner planner = HighSkillRunner.getPlanner();
planner.addHighskillAsFirst(new GetUp());

valid = true

Podmienka 2. nebola zmenená, podmienka 3. po konzultácií zatiaľ nezmenená. Po opísaných zmenách bolo plánovanie trajektórie agenta funkčné, testovaním zmien bolo treba vyhodnotiť jeho efektivitu. Podmienku 3. navrhujem prekonzultovať a upraviť v budúcnosti.

5.1.36. Vypracovanie jednoduchých TODO

Jedná sa o dve nasledovné TODO:

Class: sk.fiit.jim.agent.trajectory.Obstacles.java

(72, 12) FIXME: Nova verzia ma v nastaveniach Agent Radius 0.4. Je otazne z coho je tato hodnota 0.2 urcena.

Class: sk.fiit.testframework.ui.GameView.java

(324, 50) (337, 51) (350, 32) (364, 35) (367, 35) TODO: treba zapocitat odchylku

Prvé TODO bolo len otestované, pričom agent sa správa veľmi podobne pre AgentRadius = 0.4 ako pre agentRadius 0.2 Ani po opakovanom spustení a testovaní agenta nebol spozorovaný žiadny rozdiel, tak sa hodnota ponechala na 0.2. Druhé TODO bolo z časti už vypracované, takže sa len pridala tolerancia podľa toho ako bola počítaná v iných triedach (LinePatternRecognition).

5.1.37. Oprava parametrov spúšťania agenta

Zmenami v organizácií knižníc v rámci konvertovania projektu do systému Maven bola spôsobená nefunkčnosť pridávania agenta v TestFrameworku. Pôvodne sme predpokladali, že je chyba v rôznych verziách knižníc, ktoré neboli v prvej verzii nastavenia zjednotené. Po ich zjednotení však problém pretrvával.

Debugovaním sa zistilo, že pridávanie agenta sa deje spúšťaním nového procesu mimo TestFrameworku, kde parametre vrátane ciest ku knižniciam sú fixne zadané v súbore sk/fiit/testframework/init/default.properties. Po opätovnom pridaní knižníc do priečinku /lib bolo pridávanie stále nefunkčné.

Pri spustení pôvodného príkazu:

java -classpath ../RoboCupLibrary/bin;bin;lib/aspectjrt.jar;lib/bsf.jar;lib/jruby-complete-1.4.0.jar;lib/commons-logging-1.1.jar;lib/commons-net-2.2.jar sk.fiit.jim.init.Main

v CMD bolo zistené, že niektoré knižnice stále neboli obsiahnuté v danom príkaze a bolo ich potrebné doplniť. Po doplnení sa dal takýmto spôsobom agent spustiť v CMD ale Testframework stále ukazoval chyby spôsobené nefunkčným kódom. Chyby v kóde boli opravené.

Po opísaných zmenách bolo pridávanie agenta znova funkčné, avšak je zázrak, že pri implementovanom riešení predošlými tímami a jeho kvalite vôbec niekedy fungovalo. Odporúčali by sme riešenie prehodnotiť a implementovať nanovo.

5.1.38. Opis pridávania agenta

Pri konvertovaní projektu do Maven profilu bolo znefunkčnené pridávanie agenta v TestFrameworku. Zistili sa však, že to bolo spôsobené nesprávnymi parametrami v príkaze spúšťania. Parametre boli opravené a dohodlo sa, aby bola táto logika lepšie opísaná na Wiki.

Spúšťanie agenta je obslúžené metódou *startAgent()* v triede sk.fiit.testframework.communication.agent.AgentManager. Funguje na základe spustenia štandardného cyklu agenta v novom Java procese. Nastavenia procesu so všetkými parametrami sa nachádza v súbore sk/fiit/testframework/init/default.properties. Dôležité je hlavne nastavenie *robocup.player.command* kde sú aj cesty k potrebným knižniciam. Tie musia presne sedieť, aby ich proces vedel nájsť. Momentálne je funkčná forma (pre Windows):

java -classpath ../RoboCupLibrary/bin;bin;lib/aspectjrt.jar;lib/bsf.jar;lib/jruby-complete-1.4.0.jar;lib/commons-logging-1.1.jar;lib/commons-net-2.2.jar;lib/commons-math3-3.6.1.jar;lib/reflections-0.9.9-RC1.jar;lib/google-collections-0.9.jar;lib/javassist-2.6.jar sk.fiit.jim.init.Main

Knižnice .jar musia byť uložené v priečinku *lib* v adresáre modulu Jim. Ak budú pridané do kódu nové knižnice na spustenie agenta, je ich takisto potrebné pridať aj do týchto parametrov, aby bolo pridávanie agenta z Testframeworku funkčné. Na základe popisu by mali byť v budúcnosti problémy s aktualizáciou a opravami tejto logiky odstránené. Popis sa pridal na Wiki, aby bol dostupný aj pre ďalšie tímy.

5.1.39. Úloha move to RoboCupLibrary

Jedná sa o vypracovanie idenfitikovaného TODO. V Triede sk.fiit.jim.agent.AgentInfo sa nachádza metóda *isInHalfPlane*(), ktorá by mala byť presunutá do modulu RoboCulLibrary. Metóda vypočitáva, či bod typu *Vector3D* spadá do polroviny.

Metóda bola presunutá do triedy sk.fiit.robocup.library.geometry.Vector3D a bola upravená, aby používala súradnice z objektu, nie zo vstupného vektora.

Presunutím bolo splnené TODO, ktoré bolo vytvorené zrejme z dôvodu, že sa z abstratkného hľadiska metóda nehodila do danej triedy.

5.1.40. Referencie na iné kontrolery

Jedná sa o vypracovanie idenfitikovaného TODO. V Triede *sk.fiit.testframework.ui.controllers.TabTournamentController* sa nachádza metóda *updateTournamentTab*(), ktorá by mala byť presunutá do triedy *sk.fiit.testframework.ui.controllers.MainFrameController*, pretože v triede *TabTournamentController* sa pomocou agregácie volajú metódy triedy *MainFrameController.*

Pôvodné znenie komentára:

„TODO: referencie na ine kontrolery - nechceme to nejak inak riesit? Navrhnuté riešenie cez MainFrameController, ktorý by obsahoval metódy ktoré by sa volali“.

Metóda bola presunutá do triedy *sk.fiit.testframework.ui.controllers.MainFrameController* a bola upravená, viď obrázok nižšie. Metóda *updateTournamentTab()* sa v triede *MainFrameController* volá v metóde *update() -> run()*, ktorá je prepísaná v danej triede.

```
public void updateTournamentTab(MeasurableInformation atStart, SimulationState ss,MeasurableInformation now,MeasurableInformation dist) {
    RobocupMonitor m = getTabTournamentController().getMonitor();

    if (getTabTournamentController().getTournamentTabbedPane().getSelectedComponent().equals(getTabTournamentController().getTabWalkStability())) {
        executeTabWalkStabilityController(atStart, ss, now, dist, m);
    }

    if (getTabTournamentController().getTournamentTabbedPane().getSelectedComponent().equals(getTabTournamentController().getTabWalkSpeed())) {
        executeTabWalkSpeedController(atStart, ss, now, dist, m);

    } else if (getTabTournamentController().getTournamentTabbedPane().getSelectedComponent().equals(getTabTournamentController().getTabKickAccuracy())) {
        executeTabKickAccuracyController(atStart, ss, now, dist, m);

    } else if (getTabTournamentController().getTournamentTabbedPane().getSelectedComponent().equals(getTabTournamentController().getTabPointAccuracy())) {
        executeTabPointAccuracyController(atStart, ss, now, dist, m);

    } else if (getTabTournamentController().getTournamentTabbedPane().getSelectedComponent().equals(getTabTournamentController().getTabPlayerTurn())) {
        executeTabPlayerTurnController(atStart, ss, now, dist, m);

    } else if (getTabTournamentController().getTournamentTabbedPane().getSelectedComponent().equals(getTabTournamentController().getTabStandUp())) {
        executeTabStandUpController(atStart, ss, now, dist, m);

    } else if (getTabTournamentController().getTournamentTabbedPane().getSelectedComponent().equals(getTabTournamentController().getTabAttackOnGoal())) {
        executeTabAttackOnGoalController(atStart, ss, now, dist, m);

    } else if (getTabTournamentController().getTournamentTabbedPane().getSelectedComponent().equals(getTabTournamentController().getTabOrientedKick())) {
        executeTabOrientedKickOnGoalController(atStart, ss, now, dist, m);
    }
}
```

Obrázok - metóda updateTournamentTab()

Presunutím bolo splnené TODO, ktoré bolo vytvorené zrejme z dôvodu, že sa z abstratkného hľadiska metóda nehodila do danej triedy. V triede *TabTournamentController* sa metóda

zakomentovala s poznámkou, že metóda *updateTournamentTab()* sa presunula do inej triedy a príznak TODO sa odobral.

5.1.41. Implementácia Agent's goal is allways left, also during second period

Agent‘s goal is allways left, also during second period je TODO, ktoré sa nachádza v triede AngetInfo.java pri premennej side. Táto premenná je inicializovaná na hodnotu LEFT, čo vyjadruje, že naša bránka je na ľavej strane, a útočíme na pravú stranu. Hodnota tejto premennej je upravovaná v triede AgentModel.java v metóde processNewServerMessaage. Upravovanie tejto hodnoty je závislé od dvoch ďalších premenných. Prvou premennou je HALF_TIME_CHANGE_SIDES, čo je booleanovská premenná vyjadrujúca, či sa počas počasu majú alebo nemajú vymeniť strany. Druhou premennou je OUR_SIDE_IS_LEFT, ktorá je inicializovaná na null, a jej hodnota môže byť upravená správou zo serveru. Ak zo server pride nastavenie premennej na true, tak side sa taktiež upraví na LEFT, ale ak pride false tak sa side upraví na RIGHT. V prípade, ak má premenná HALF_TIME_CHANGE_SIDES hodnotu true, a premenná OUR_SIDE_IS_LEFT ostane null (čo znamená, že naša strana nie je určená serverom), hodnota premennej side sa správne počas polčasu zmení z LEFT na RIGHT alebo opačne. Avšak táto zmena pre “diváka” nič neupraví, keďže z pohľadu kamery sa stale útočí z ľavej strany na pravú, pričom aj súradnice ihriska ostávajú rovnaké – naša bránka ma zápornú x-ovú súradnicu a súperová kladnú. Preto nie je jasné, či premenná nie je správne využívana, alebo sa len hodnoty prepočítavajú tak, aby všetko ostalo na prvý pohľad nezmenené.

Navrhnutý test na overenie funkcie strán by bol vytvorenie dvoch agentov v dvoch proti sebe hrajúcich tímoch a testovaním funkcie agentov pre rôzne nastavenia premennej side.

Navrhnutý test nebol vykonaný, z dôvodu problémov s TestFrameworkom.

5.1.42. Implementácia try to create solution with reusable Singleton Factory

Jedná sa o vypracovanie identifikovaného TODO. Predošlý tím navrhol pri vyberaní pohybov chodenia v rámci triedy sk.fiit.jim.agent.highskill.move.MovementSkills. Pri volaní sa vždy vytvárali nová inštancia pohybu a to zrejme nebolo úplne efektívne, tím preto navrhol implementovať Singleton Factory, ktorý by vždy vracal tú istú inštanciu.

Vytvorená trieda sk.fiit.robocup.library.SingletonFactory (metóda getInstance()) dokáže vytvoriť „singleton“ z ľubovoľného typu, ktorý vchádza ako parameter. Potom sa táto trieda vloží do mapy a vždy keď je potrebná, sa z mapy vyberie. Ak sa v mape nenachádza, vytvorí sa nová. Metóda sa potom volá primárne v súvislosti s podtriedami typu Walk.

Pri testovaní však bolo odhalené, že pri konkrétnom použití na podtriedy typu Walk hráč nefunguje správne. Z nezisteného dôvodu je pri volaní týchto tried vždy potrebná nová inštancia. Pravdepodobne je to spôsobené zmenou atribútov, ktoré pre správnu funkčnosť musia byť vždy v defaultnom stave. Bližšiu príčinu sa odhaliť nepodarilo. Trieda SingletonFactory v kóde môže ostať, avšak nie je možné ju použiť na prvotne zamýšľaný účel.

5.1.43. Zmena spartakiády na zmysluplný pohyb

Jim sa vedel v rozhodovacom strome dostať do posledného „else“ stavu, v ktorom nevedel čo má vykonať a tak sa vrátil do čias socializmu a začal sa pripravovať na starú dobrú spartakiádu. Návrat do minulosti mu však nepomohol v tom, aby sa v súčasnosti zlepšil vo futbale. Preto sme museli prísť na riešenie, ktoré mu pomôže prestať snívať, spamätať sa a začať hrať futbal. Ako sme to vyriešili sa dočítate v kapitole vypracovanie.

Prvou podstatnou časťou bolo zistiť prečo sa vôbec Jim do tohto stavu dostane a aký lowskill by bolo najlepšie v danej situácii využiť. S dlhodobého skúmania situácie Jima a dôvodu prečo sa vôbec začne vykonávať tento zdraviu užitočný pohyb sme zistili, že najvhodnejšie bude namiesto priameho LowSkillu vykonávať HighSkill Localize. Tento skill Jimovi umožní obnoviť svoje znalosti o lokácii a pomôže mu so svižným návratom do hry. Na toto sme museli však upraviť/pridať viaceré xml súbory a opraviť 2 triedy.

Ovplyvnené xml súbory sú head_down.xml, head_left.xml, head_right.xml a head_left_and_right.xml.

Ovplyvnené boli tieto triedy:

Walk.java – táto trieda obsahovala rozhodovací strom v ktorom san a konci dostal Jim k pokynu vykonávania spartakiády. Toto bolo napravené pridaním HighSkillu “Localize” do zoznamu vykonávaných HighSkillov na začiatok a následné vyvolanie LowSkillu “rollback”.

Localize.java – tento HighSkill bol do vysokej miery upravený – nastalo pridanie možnosti pozrieť sa dole. Taktiež bola upravená rozhodovacia štruktúra. Pridaním rozhodnutia o pozrení dole bolo pre efektívnosť vhodné pridať metódu, ktorá resetuje histótiu vykonania pohybov hlavou pri zmene polohy.

Jim už nevykonáva spartakiádu, no snaží sa lokalizovať.

5.2. Testovanie

Táto kapitola zahŕňa opis všetkých testovaní, ktoré sa vykonali od začiatku semestra na overenie jednotlivých implementácií.

5.2.1. Testovanie úlohy „Pozrieť sa na AgentModel“

Trieda sk.fiit.jim.agent.models bola podľa yourkit výpočtovo náročnejšia. Úlohou bolo pozrieť sa na túto triedu a zrýchliť metódy, ktoré sa dajú. Počas implementácie sa zmenilo logovanie triedy, ktoré sa vykonávalo, aj keď nebolo treba (bolo vypnuté). Pomocou yourkit sme 10x odmerali hodnoty danej triedy a tie porovnali.

Číslo merania	Pred zmenou	Po zmene
1	51%	29%
2	46%	30%
3	45%	29%
4	44%	29%
5	43%	31%
6	43%	30%
7	43%	30%
8	43%	29%
9	42%	29%
10	42%	29%
Priemer	44,2%	29,5%

Tabuľka 32 Výsledky merania AgentModel

Zlepšenie o (44,2%-29,5%) 14,7%. Celkovo sa zlepšila náročnosť triedy o 14,7%.

5.2.2. Testovanie úlohy „Pozrieť sa na LowSkills“

Bolo implementované vypnutie logovania ošetrením podmienkou i*f(Setting.getBoolean(„lowSkillsLogger“))*.

1. 5x meranie v yourkit po zmene implementácie.

Poradové číslo merania	Hodnota výpočtovej náročnosti danej triedy v yourkit
1	1%
2	3%
3	2%

4	2%
5	1%
Priemer	1,8%

Tabuľka 33 Výsledky merania LowSkills

Pôvodnú hodnotu 7% sa podarilo znížiť na 1,8%, čo je zlepšenie o 5,2%. Implementácia logovania v LowSkills bola otestovaná. Zlepšila sa výpočtová náročnosť o 5,2%.

5.2.3. Testovanie úlohy pridanie viac vlákien do spracovnia komunikácie

Christopher implementoval vlákna do spracovania komunikácie, čo má zabezpečiť rýchlejšiu komunikáciu. Testovať takúto implementáciu sa dá iba meraním. Na meranie sa využil softvér Yourkit.

1. Odmerať 10x rýchlosť komunikácie pred implementovaním zmien
2. Odmerať 10x rýchlosť komunikácie po implementovaní zmien

Číslo merania	Pred zmenou implementácie	Po zmene implementácie
1	48%	45%
2	49%	41%
3	48%	39%
4	47%	36%
5	46%	38%
6	45%	37%
7	51%	35%
8	42%	34%
9	58%	34%
10	47%	45%
priemer	48,1%	38,4%

Tabuľka 34Výsledky merania pri testovaní priadnia viac vlákien

Zmenou implementácie sme zlepšili komunikáciu Jima v priemere o 9,7%. Obrazovky merania z Yourkitu boli nahrané na dokumentový server. Avšak chovanie sa Jima zhoršilo, čo spôsobujú pakety, ktoré sa miešajú z dôvodu, že sú vo vláknach. Z tohto dôvodu je vytvorený nový task 9934 - Miešanie pakiet a celkové chovanie sa Jima po implementácií vlákien.

5.2.4. Testovanie synchronizácie komunikačného vlákna

Po pridaní vlákien bolo potrebné správne synchronizovať komunikačné vlákno. Po pridaní vlákien bez synchronizácie robot viac padal a po niektorých pádoch sa nevedel postaviť, čo bolo spôsobené nesprávnou synchronizáciou komunikačného vlákna.

2. Pozorovanie správania sa robota.
3. Vyvodenie záveru.

Robota sme pozorovali dlhšiu dobu, identifikovali sme menej pádov. Po každom páde sa postavil.

Aktuálne robot menej padá ako pred synchronizáciou komunikačného vlákna. Po každom páde sa postavil. V ďalších šprintoch sa treba viac zamerať na testovanie a vychytanie chýb pri komunikácií. Preto sme pridali novú user story: Testovanie súčasného stavu.

5.2.5. Testovanie odstránenia posielania správ a výpisov do logov

Bolo implementované vypnutie posielania správ do testframeworku a vypnuté logovanie.

1. Overiť, že logovanie sa neukladá, ak je vypnuté v agent_model.
2. Overiť, že logovanie sa ukladá, ak je zapnuté v agent_model.

Testovacie scenáre boli spustené a overené. Oba testovacie prípady boli na 100% úspešné. Implementácia zmeny posielania správ bola otestovaná. Testy boli na 100% úspešné.

5.2.6. Testovanie funkcie isPlus

Cieľom úlohy bolo implementovať funkciu isPlus, ktorá zisťuje, či majú dve úsečky spoločný priesečník v tvare +. Na jej základe môže hráč lepšie určiť svoju polohu – podľa relatívnych súradníc, ktoré dostane od servera vypočíta, či ide o tvar +.

Základné testovanie vypracovaných metód sa nachádza v triede LinePatternRecognitionTest. Pre účely testovania novej metódy sú v teste vytvorené nové objekty čiar, ktoré netvoria + a také, ktoré tvoria +. Správnosť výsledku metódy je potom skontrolovaná porovnaním s očakávanou hodnotou. Príklad volania:

assertEquals(0, isPlus(lines.get(6), lines.get(7)));

Pomocou implementovanej metódy *isPlus()* je možné zistiť, či dve úsečky majú priesečník v tvare +, ktorý je pre vyhodnotenie polohy hráča relevantný a teda je možné ju použiť na zlepšenie orientácie hráča na ihrisku. Unit testy sú úspešné a môžeme považovať implementáciu funkcie za dokončenú.

5.2.7. Testovanie funkcie isT

Cieľom úlohy bolo implementovať funkciu isT, ktorá zisťuje, či majú dve úsečky spoločný priesečník v tvare T. Na jej základe môže hráč lepšie určiť svoju polohu – podľa relatívnych súradníc, ktoré dostane od servera vypočíta, či ide o tvar T.

Základné testovanie vypracovaných metód sa nachádza v triede LinePatternRecognitionTest. Pre účely testovania novej metódy sú v teste vytvorené nové objekty čiar, ktoré netvoria T a také, ktoré tvoria T. Správnosť výsledku metódy je potom skontrolovaná porovnaním s očakávanou hodnotou. Príklad volania:

assertEquals(0, isT(lines.get(6), lines.get(7)));

Pre potreby testovania bola pridaná funkcia na menenie endFlagu – *setEndFlag1(Boolean bool)* a *setEndFlag2(Boolean bool)*.

Pomocou implementovanej metódy *isT()* je možné zistiť, či dve úsečky majú priesečník v tvare T, ktorý je pre vyhodnotenie polohy hráča relevantný a teda je možné ju použiť na zlepšenie orientácie hráča na ihrisku. Unit testy sú úspešné a môžeme považovať implementáciu funkcie za dokončenú.

5.2.8. Opis testovania funkcie justLinesSeen

Počas práci na projekte sme sa zamerali aj na novovzniknutú funkciu justLineSeen, konkrétne priesečníkmi čiar vytvárajúcich vzory ‚T' a ‚+' alebo iných nežiadúcich vzorov. Výstupom funkcie sú relatívne pozície bodov čiar reprezentujúcich vzory 'T' a '+'. Na základe týchto súradníc budeme neskôr môcť otestovať správne vnímanie vzorov 'T' a '+' agentom.

Prvotne bolo dôležité uvedomiť si, že agent, ktorý sa nachádza v poli (pole je opísane osami X a Y) môže byť umiestnený v rôznych bodoch a jeho percentuálny vnem nemusí vnímať všetky dostupne kontrolné resp. Riadiace body, ktoré by mal vedieť rozpoznávať. Vytvorili sme preto 4 testovacie scenáre, kedy sme agentovi určili fixnú polohu v poli smerom na konkrétny vzor. Z tejto polohy sme neskôr vypočítali relatívne vzdialenosti bodov (z ktorých sa skladajú identifikované vzory) od polohy agenta, čo je naším primárny výstupom pre testovanie.

5.2.8.1. Body priesečníkov a vzorov

Postup vytvárania testovacích scenárov bol pomerne nezložitý. Spočíval v:

• Umiestnení agenta do poľa smerom na 1 zo vzorov

• Nastavenie kamery na rovnaký bod v poli

• Natočenie kamery na konkrétny vzor

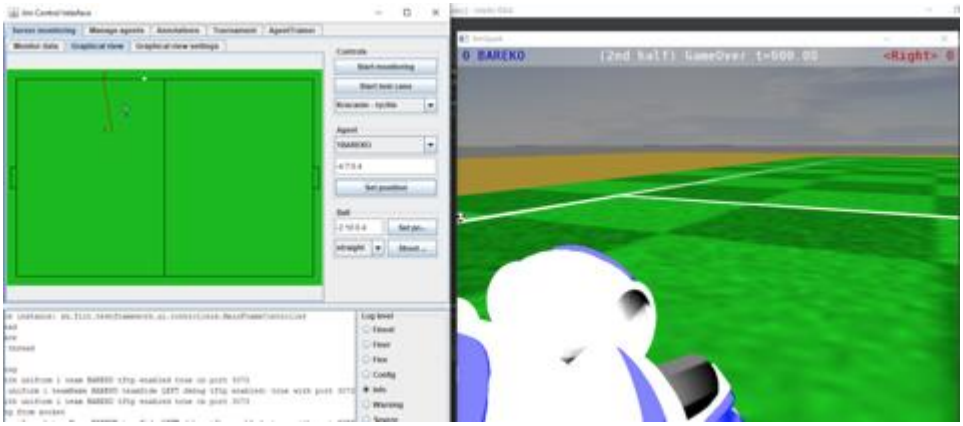
• Určenie resp. identifikovanie bodov na vzory

Identifikovanie bodov vzoru sme riešili prednastavením lopty na stredovú čiaru rozpoľujúcu ihrisko na 2 polovice. Po tom ako sme natočili smer kamery na konkrétny vzor sme loptu posúvali po osi aby sa vždy nachádzala v zábere kamery (viď. obrázky nižšie). Tento postup bol zvolený len na autentickejšie určenie bodov identifikovaného vzoru. Samozrejme si uvedomujeme, že agenta dokáže vnímať až 120°uhol.

5.2.8.2. Vzor horného T

Ihrisko sa nachádza na **Z-ovej** osi vo výške **0,4** v každom prípade, tým pádom ďalej tento údaj nebudeme uvádzať.

V tomto prípade sme určili nasledujúcu **pozíciu agenta**: x: -4, y: 7

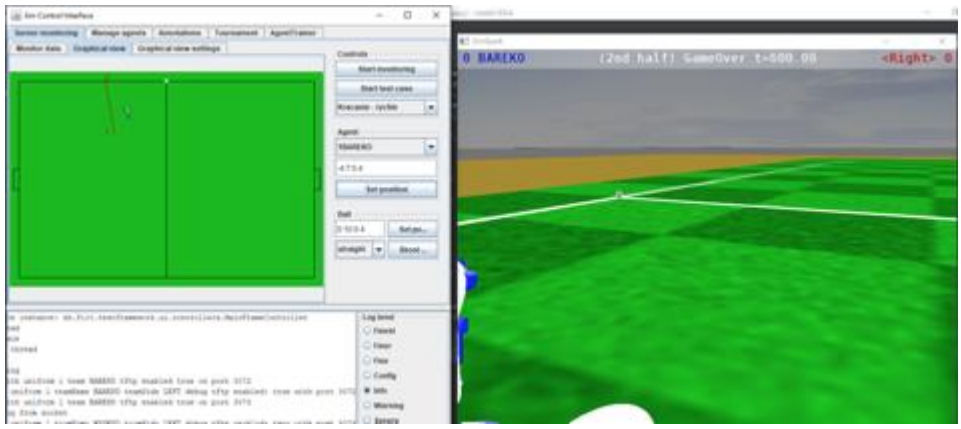


Obrázok 50 Poloha ľavého bodu horného vzoru T

Poloha ľavého bodu horného vzoru T: x: -2, y: 10

Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

x: (od -4 po 7) = 2, y: (od 10 po 7) = 3

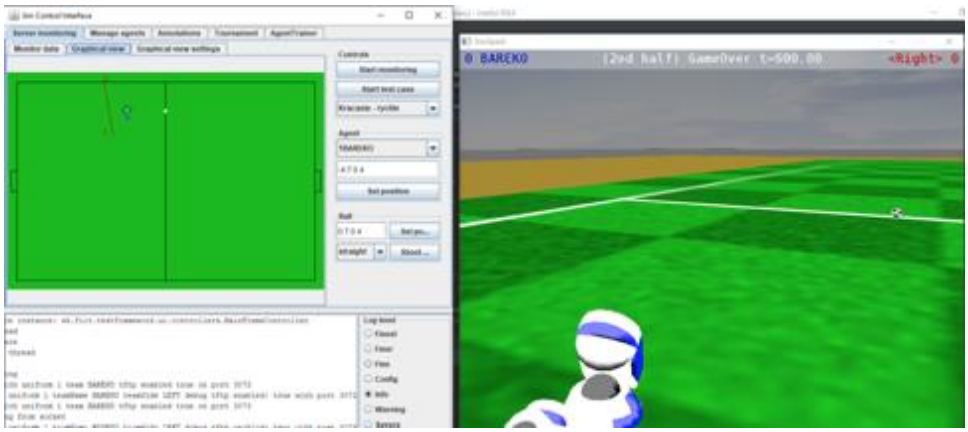


Obrázok 51 Poloha stredného bodu horného vzoru T

Poloha stredného bodu horného vzoru T: x: 0, y: 10

Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

x: 4, y: 3

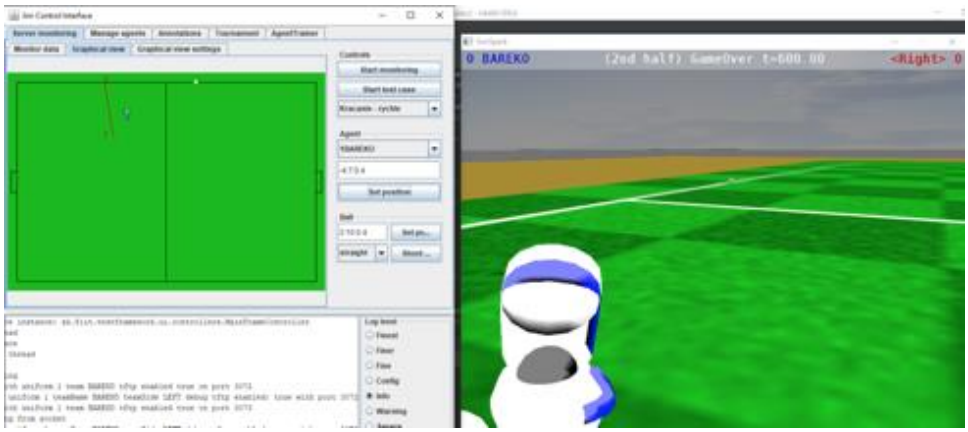


Obrázok 52 Poloha dolného bodu horného vzoru T

Poloha dolného bodu horného vzoru T: x: 0, y: 7

Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

x: 4, y: 0



Obrázok 53 Poloha pravého bodu horného vzoru T

Poloha pravého bodu horného vzoru T: x: 3, y: 10

Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

x: 7, y: 3

5.2.8.3. Vzor dolného T

V tomto prípade sme určili nasledujúcu **pozíciu agenta**: :x: -4, y: -7

- Poloha pravého bodu dolného vzoru T: x: 3, y: -10

  Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

  x: 7, y: -3

- Poloha ľavého bodu dolného vzoru T: x: -3, y: -10

  Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

  x: 1, y: -3

- Poloha stredného (priesečník) bodu dolného vzoru T: x: 0, y: -10

  Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

  x: 4, y: -3

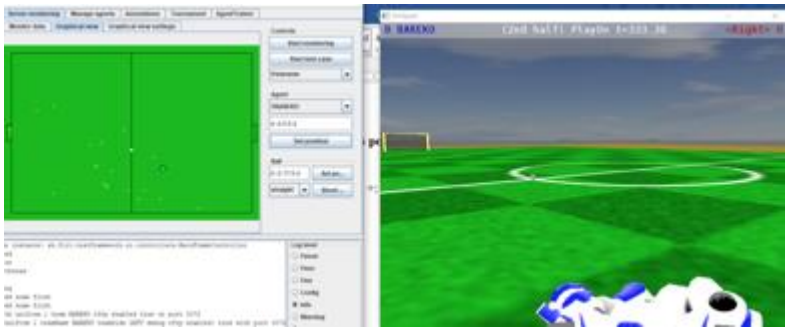
- Poloha horného bodu dolného vzoru T: x: 0, y: -7

  Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

  x: 4, y: 0

5.2.8.4. Vzor dolné +

V tomto prípade sme určili nasledujúcu **pozíciu agenta**: :x: 4, y: -4.5

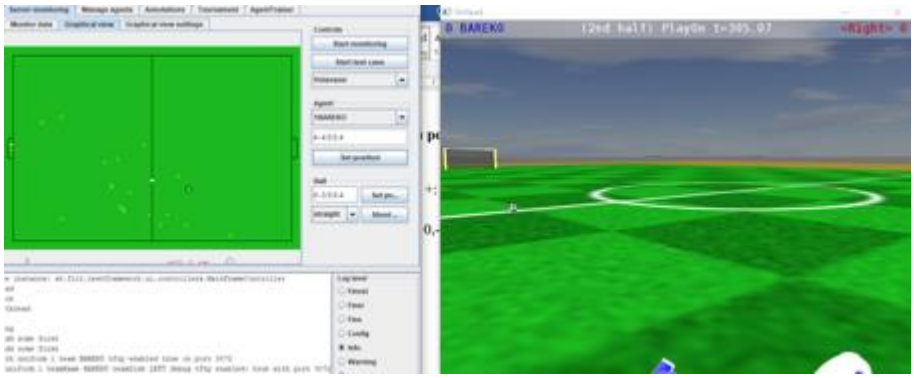


Obrázok 54 Poloha stredového bodu dolného vzoru +

Poloha stredového bodu dolného vzoru +: x: 0, y: -2.17

Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

x: -4, y: 2.33

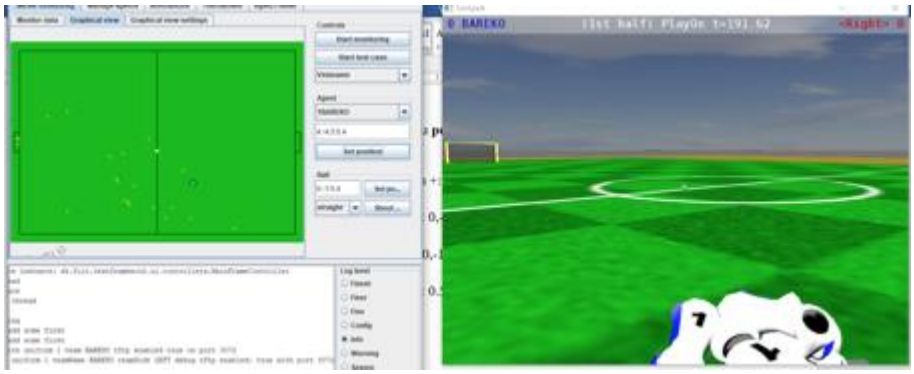


Obrázok 55 Poloha dolného bodu dolného vzoru +

Poloha dolného bodu dolného vzoru +: x: 0,y: -3.5

Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

x: -4, y: 1

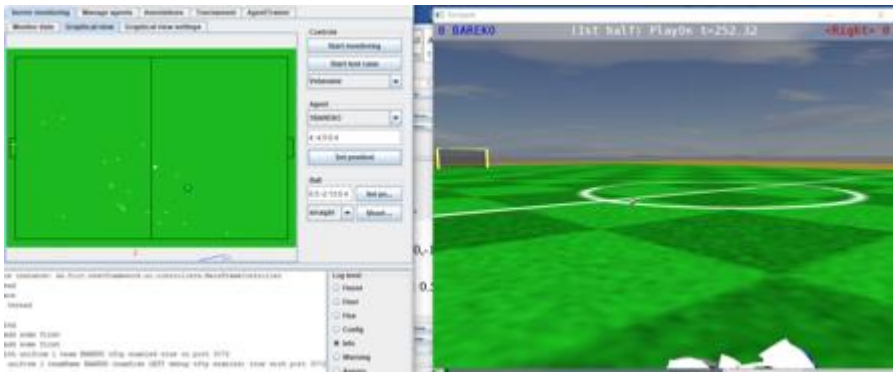


Obrázok 56 Poloha horného bodu dolného vzoru +

Poloha horného bodu dolného vzoru +: x: 0,y: -1

Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

x: -4, y: 3.5

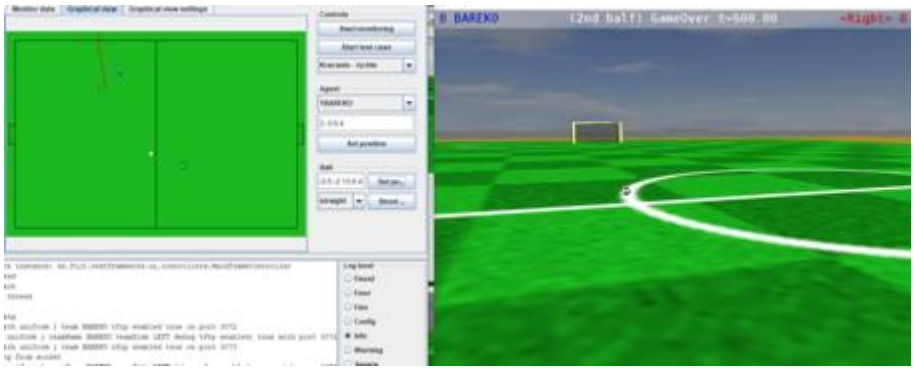


Obrázok 57 Poloha pravého bodu dolného vzoru +

Poloha pravého bodu dolného vzoru +: x: 0.5,y: -2.13

Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

x: -3.5, y: 2.37



Obrázok 58 Poloha ľavého bodu dolného vzoru +

Poloha ľavého bodu dolného vzoru +: x: -0.5,y: -2.13

Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

x: -4.5, y: 2.37

5.2.8.5. Vzor horné +

V tomto prípade sme určili nasledujúcu **pozíciu agenta**: :x: 4, y: 4.5

- Poloha stredového bodu (priesečník) horného vzoru +: x: 0, y: 2.17

  Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

  x: -4, y: 2.33

- Poloha horného bodu horného vzoru +: x: 0,y: 3.5

  Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

  x: -4, y: -1

- Poloha dolného bodu horného vzoru +: x: 0,y: 1

  Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

  x: -4, y: -3.5

- Poloha pravého bodu horného vzoru +: x: 0.5,y: 2.13

  Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

  x: -3.5, y: -2.37

- Poloha ľavého bodu horného vzoru +: x: -0.5,y: -2.13

  Vyrátaná relatívna vzdialenosť od polohy agenta k danému bodu:

  x: -4.5, y: -6.63

Výsledkom opisu testovania funkcie JustLineSeen sú súradnice vzorov v bodoch a relatívne vzdialenosti bodov od polohy agena pre konkrétne štyri testovacie scenáre. Pre vzory „T“ sú to vždy štyri body a pre vzory „+“ je to päť bodov. Pomenovanie pozície bodov je určené podľa mapky poľa v graphical view v Jim Control Interface. Pozícia bodov bola určená umiestnením lopty na konkrétny priesečník. Z pohľadu polohy agenta bola následne vypočítaná relatívna vzdialenosť ku každému bodu.

5.2.9. Test implementácie JustLinesSeen

Doimplementovala sa funkcia justLinesSeen, ktorá sa volá, ak agent vidí iba čiary. Bolo treba na základe nich identifikovať priesečníky a pomocou funkcií určujúcich vzor čiar zistiť, o ktorý bod na ihrisku sa jedná a na základe toho mu priradiť kombináciu relatívnych a absolútnych súradníc.

Boli pridané unit testy na overenie správnosti implementácie. Unit testy prebehli s 100% úspešnosťou.

Okrem unit testov bol agent bol viackrát pustený a otestovaný. Nezistilo sa, že by nová implementácia mala na jeho správanie negatívny vplyv.

5.2.10. Test implementácie Effectors into HashMap

Boli potrebné vykonať zmeny dátových štruktúr v triedach: JoiuntPlacement.java, Phase.java a SkillsFromXmlLoader.java.

Testovací scenár:

- Pustenie agenta
- Overenie funkčnosti agenta

Agent bol viackrát pustený a otestovaný. Nezistilo sa, že by nová implementácia mala na jeho správanie vplyv.

5.2.11. Testovanie implementácie „Implementácia singleton vzoru v SkillsFromXmlLoader“

Bolo potrebné zaviesť singleton, aby sa pracovalo s nanajvýš jednou inštanciou triedy. Zmeny boli vykonané v triedach: KickTestHighSkill, JimWindowController, Main, SkillsFromXmlLoader, TestFrameworkMain, LowSkillTest, SkillsFromXmlLoader.

Testovací scenár:

- Pustenie agenta
- Overenie funkčnosti agenta

Agent bol viackrát pustený a otestovaný. Nezistilo sa, že by nová implementácia mala na jeho správanie vplyv.

5.2.12. Test implementácie „opraviť prípad, keď je size 0“

V metóde updatePosition bola implementovaná podmienka. Po implementácií úlohy nasledovalo jeho testovanie.

Testovací scenár:

- Pustenie agenta, aby sa dostal do takého stavu, kde sa vyvolá metóda updatePosition
- Overenie funkčnosti agenta

Agent bol viackrát otestovaný. Po zavolaní danej metódy agent pokračoval v hre a metóda už nehodila exception. Na správanie agenta implementácia metóda nemá výrazný vplyv.

5.2.13. Test implementácie vymazania nepoužitých metód časť 2

Cieľom úlohy bolo vymazať nepoužité a nepotrebné metódy. Slúži to na sprehľadnenie projektu, avšak funkcionalita Jima sa nemá zmeniť. Tento dokument opisuje test, ktorým sa overila funkčnosť projektu.

Boli modifikované triedy:

Jim\src\sk\fiit\jim\agent\models\TacticalInfo.java

Jim\src\sk\fiit\jim\agent\models\WorldModel.java

Jim\src\sk\fiit\jim\agent\parsing\Perceptors.java

RoboCupLibrary\src\sk\fiit\robocup\library\geometry\Point2D.java

TestFramework\src\sk\fiit\testframework\annotator\serialization\XMLcreator.java

TestFramework\src\sk\fiit\testframework\communication\robocupserver\RobocupServer.java

TestFramework\src\sk\fiit\testframework\trainer\testsuite\TestCaseHelper.java

Jim\src\sk\fiit\jim\decision\situation\NoOneHasBall.java

Jim\src\sk\fiit\jim\annotation\data\XMLCreator.java

Jim\src\sk\fiit\jim\agent\parsing\SeePerceptor.java

TestFramework\src\sk\fiit\testframework\annotator\serialization\XMLparser.java

Niektoré nájdené „nepoužívané“ metódy vymazané neboli.

Testovací scenár:

1. Nahodenie zmien
2. Pustenie jima
3. Overenie funkčnosti
4. Pustenie TestFramework
5. Overenie funkčnosti

Na správanie agenta zmeny v implementácií nemali negatívny vplyv. Nezistilo sa, že vymazané metódy ovplyvnili funkcionalitu.

5.2.14. Test implementácie opravy parametrov spúšťania agenta

Po prekonvertovaní projektu do Mavenu sa nedal spustiť TestFramework. Po analýze a debugovaní sa zistilo, že niektoré parametre boli zle nastavené. Túto úlohu bolo potrebné znovu implementovať, nakoľko prvá implementácia neprešla testom.

Testovací scenár:

- Nahodenie zmien
- Pustenie TestFrameworku
- Overenie funkčnosti

Po spustení kódu so zmenami je TestFramework funčný. TestFramework bol pustený, agent sa už načíta a TestFramework funguje.

5.2.15. Test implementácie referencii na iné kontrolery

Dokument opisuje test implementácie referencovania na iné kontrolery. Jedná sa o úlohu identifikovaného ako TODO. V rámci implementácie bola presunutá metóda do inej triedy.

Testovanie prebiehalo stanovením testovacieho scenára a opisom výsledkov.

Testovací scenár:

- stiahnutie zmien,
- overenie funkčnosti jima,
- opis výsledkov.

Agent je rovnako funkčný ako pred zmenou implementácie. Implementáciu hodnotím ako úspešnú a nemá negatívny vplyv na funkčnosť projektu.

5.2.16. Testovanie implementácie zmena spartakiády na zmysluplný pohyb

Jim sa vedel v rozhodovacom strome dostať do posledného „else“ stavu, v ktorom nevedel čo má vykonať a tak sa vrátil do stavu, čo sme nazvali spartakiáda. Tento problém bol vyriešený tak, že v tomto stave namiesto priameho LowSkillu vykonáva HighSkill Localize. Tento skill Jimovi umožní obnoviť svoje znalosti o lokácii a pomôže mu so svižným návratom do hry.

Ovplyvnené boli tieto triedy:

Walk.java – táto trieda obsahovala rozhodovací strom v ktorom san a konci dostal Jim k pokynu vykonávania spartakiády. Toto bolo napravené pridaním HighSkillu “Localize” do zoznamu vykonávaných HighSkillov na začiatok a následné vyvolanie LowSkillu “rollback”.

Localize.java – tento HighSkill bol do vysokej miery upravený – nastalo pridanie možnosti pozrieť sa dole. Taktiež bola upravená rozhodovacia štruktúra. Pridaním rozhodnutia o pozrení dole bolo pre efektívnosť vhodné pridať metódu, ktorá resetuje histótiu vykonania pohybov hlavou pri zmene polohy.

1. Stiahnutie a build nových zdrojových súborov
2. Pustenie agenta
3. Overenie funkčnosti implementácie

Po implementácií sa agent nedostal do stavu spartakiády a vždy sa vedel zorientovať na ihrisku.

5.3. Zhrnutie práce a odporúčania pre ďalšie tímy

Tím WAWOT 2018/2019 úspešne splnil primárne stanovené ciele projektu inteligentného robotického hráča futbalu. Spoločnou prácou, vzájomným učením sa, dopĺňaním svojich schopností a vedomostí sme dotiahli výsledok ďaleko za hranicu, po ktorú by sa ktokoľvek z nás vedel dostať vlastnými silami.

5.3.1. Opis budúcnosti projektu

Projekt robotického hráča futbalu sa aktuálne nachádza v stave, v ktorom množstvo navrhovaných implementačných úloh nie je dokončených alebo implementovaných. V prvom rade navrhujeme dokončiť zostávajúce úlohy, ktoré sme nestihli dokončiť v rámci TP 2018/2019. Tieto úlohy sú nasledovné:

1. implement calculation of opponent position (č. úlohy: 10839)
2. a deviation has to be considered (č. úlohy: 10841)
3. implement computation with rotation agent to ball (č. úlohy: 10842)
1. accelerometer is relative to torsoPosition, not centerOfMass (č. úlohy: 10852)

Ďalej z našich dokumentácii a analýz vyplýva, že problematiky ako zero-moment point, kalman filter, euklidovské matice a ich transformácie neboli implementované ale iba zanalyzované. Dokumentáciu k predchádzajúcim problematikám sa nachádzajú na wiki stránke projektu. Navrhujeme ich vhodne zapracovať do projektu. Ďalej navrhujeme nasledujúcim tímom naštudovať zmeny v projekte, ktoré sme implementovali, pretože ďalšia optimalizácia je vítaná. Metódy na rozpoznanie vzorov na ihrisku môžu byť optimalizované. Môže byť aj pridané mapovanie videných koncových bodov čiar na základe absolútnych súradníc identifikovaných bodov, po vzore funkcie AgentPositionCalculator.oneFixedObjectSeen(). V súvislosti s koordináciou hráča dôrazne odporúčame vylepšiť pohyb a stabilitu chôdze hráča, pretože príliš často krát padá a nedokáže súvisle prejsť dlhšiu vzdialenosť bez toho, aby spadol na zem. Z predchádzajúceho bodu vyplýva, že robotický hráč trpí optimalizačnými chybami.

Navrhujeme prejsť celý kód hráča a optimalizovať jednotlivé jeho funkcionality. Optimalizácia alebo preprogramovanie existujúcich funkcionalít s novou myšlienkou a pochopením by mala viesť k vylepšeniu celkového výkonu hráča. Optimalizácie je nevyhnutá hlavne z dôvodu, že hráč na slabších počítačoch nedokáže promptne reagovať na okolnosti a správy zo servera, a ako bolo aj vyššie spomenuté tak príliš často krát padá na zem.

Počas práce na projekte sme natrafili na veľa nezmyselných riešení predošlých tímov, ktoré majú za následok nesprávne fungovanie hráča na ihrisku. Je potrebné prejsť celý kód, triedy a metódy s účelom analýzy správnosti riešenia. Mnohokrát sa hráč správa nepredvídane a takéto prípady je potrebné čím skôr odstrániť. Ďalej navrhujeme preštudovať celý TestFramework a spraviť k nemu dokumentáciu, pretože presne nevieme na čo sa používa a čo vlastne robí mnoho jeho funkcionalít. Otázne je aj to, či funkcionality v TestFramework-u sú vôbec správne naprogramované, nakoľko pri zbežnom prejdení kódu bolo objavených viacero pochybných/nelogických implementačných štruktúr.

Ďalším tímom odporúčame kvalitne dokumentovať všetku prácu na projekte pre ďalšiu generáciu po nich a čo je najdôležitejšie všetko zverejniť na wiki stránke projektu.

6. Záver

Na tomto projekte sme pokračovali v práci na agentovi, tam kde minulý tím skončil. Myslíme si, že naša práca na tomto projekte bola prínosná. Každým rokom sa študenti snažia projekt robotický futbal zlepšiť a myslíme si, že aj náš tím prispel k jeho zlepšeniu.

Priamu prácu na projekte predchádzala analýza už existujúcich riešení a dôkladné preštudovanie samotného projektu. Analýza bola kľúčovým krokom pre čo najlepšie pochopenie fungovania robotického agenta. Po pochopení princípov sme dokázali implementovať nové funkcionality, ktoré vedú k zlepšeniu orientácie hráča na ihrisku. Implementovali sme metódy na rozpoznávanie jednotlivých bodov na ihrisku, čím hráč dokáže presnejšie určiť svoju polohu v rámci ihriska. Vysporiadali sme sa a opravili sme chyby v projekte, ktoré boli spravené predošlými tímami. Zistili sme, že v projekte je hneď zopár závažných chýb resp. problémov, ako napríklad otočené osí x a y. Žiaľ počas práce na projekte sme natrafili na ďalšie skryté chyby a nedostatky a tak naša práca na projekte nabrala charakter opravovania chýb a zdokonaľovania. Popritom sme doimplementovali alebo refaktorovali ďalšie časti kódu, ktoré boli označené príznakom TODO. Tieto úlohy predstavovali opravu kódu, vylepšenie kódu, doprogramovanie nových funkcionalít či oprava existujúcich funkcionalít. Podarilo sa nám vyriešiť skoro všetky úlohy značené takýmto príznakom. Z dôvodu časovej tiesni a konci letného semestra sme sa ďalej nepúšťali do riešenia posledných komplikovanejších resp. ťažších úloh.

Ďalej sa nám podarilo vylepšili výkon hráča na ihrisku formou optimalizácii tried, ktoré majú na starosť pohyb hráča na ihrisku. Odstránil sme nepoužité importy knižníc, nepotrebné metódy a triedy. Podarilo sa nám taktiež upraviť logiku spracovania taktík a stratégii čo viedlo k rýchlejšiemu rozhodovaniu sa hráča akú taktiku zvolí pri hre proti súperovi. Projekt sme taktiež prekonvertovali do Maven formátu pre jednoduchšiu manipuláciu s dependencies. Táto zmena je veľkým pozitívom pre projekt, lebo sa odstránilo mnoho súborov v projekte, ktoré zbytočné robili projekt neprehľadným.

Všetky nové implementácie sme dôkladne otestovali a výsledky testov zdokumentovali. Našťastie v prípade každého testovania sme dostali pozitívne výsledky, čo naznačuje tomu, že sme efektívne a správne postupovali pri práci na projekte.

Taktiež sa nám podarilo zanalyzovať problematiky, ktoré sú vyššieho rádu pre budúce tímy a tým pádom sme im predpripravili dokumenty s analýzou jednotlivých problematík, ktoré čakajú

na zapracovanie alebo zmenu. Problematiky ako kalmanov filter a euklidovské matice predstavujú oblasti spojené hlavne s orientáciou a rovnováhou hráča.

Dosiahli sme, že projekt je výrazne sprehľadnený a aktualizovaný a ďalšia práca na ňom bude s istotou jednoduchšia pre ďalšie tímy. Zostávajúcu prácu na projekte sme zanalyzovali a uverejnili pre ďalšie tímy na wiki stránku projektu. Na wiki sme zverejnili aj dokumentáciu implementačných úloh, ktoré sú kľúčové z hľadiska zmien v projekte.

Prílohy

Príloha A – Metodika pracovania s TFS

1. Základné informácie

Názov tasku: Metodika pracovania s TFS

Číslo tasku: 9584

2. Úvod

Na riadenie tímovej práce sme si vybrali nástroj Microsoft TFS, ktorí našim požiadavkám vyhovoval najviac. Tento dokument opisuje, ako využívať väčšinu funkcií, ktoré tímu pomáhajú pri vývoji.

3. Vypracovanie

V rámci šprintu je potrebné predovšetkým vytvárať nové user story a tasky, následne ich priradiť a potom správne presúvať medzi stavmi. Takisto je dôležité správne ukladať dokumentáciu, ktorá bola v rámci tasku vytvorená.

Na konci šprintu je potrebné user story riadne skontrolovať a uzavrieť. V prípade nedokončenia niektorej zo zadaných úloh sa user story rozdelí a priradí sa do nového šprintu.

4. Vytváranie nových itemov

Väčšinou pred začiatkom šprintu, ale aj počas je možné vytvoriť nové user story vo forme backlog itemov v záložke **Backlog items**. Mali by mať čo najunikátnejšie mená v rámci všetkých itemov vytvorených počas vývoja. Ak item tím chce do aktuálneho šprintu, je možné ho v režime **Backlog** pomocou kontextového menu presunúť (**Move to iteration**). Následne pri plánovaní bude určený effort na jeho splnenie a priorita. Po spracovaní špecifikácie user story budú preň vytvorené tasky, ktoré taktiež dostanú krátky opis, ktorý sa určí na spoločnom stretnutí na základe špecifikácie user story.

5. Priradenie taskov

Po vytvorení taskov si členovia na spoločných stretnutiach vyberajú, ktoré v priebehu šprintu vypracujú. Priradenie sa za výnimočných okolností môže neskôr aj zmeniť. Ideálne sa priradí tasku aj človek, ktorý bude robiť review prípadne testovanie. V tasku na to existujú 2 stĺpce: **Review asignee** a **Test asignee**. Pre fungovanie notifikovania do služby Slack je ale lepšie pridať na task tag s menom člena, ktorý tieto úlohy vykoná v tvare #[priezvisko]. Tak bude

zabezpečené, že priradení členovia budú dobre informovaní o stave tasku, a budú vedieť kedy môžu s review resp. testovaním začať.

6. Ukladanie dokumentácie

Ak si task žiadal vypracovanie nejakej dokumentácie a keďže TFS neumožňuje odkazovanie dokumentov v repozitári **Doc** do tasku, túto dokumentáciu je potrebné uložiť na task ako prílohu (záložka **Attachments**) a takisto aj do repozitára do príslušnej kategórie. Ak kategória neexistuje, je možné ju vytvoriť. Názvy dokumentov budú v tvare [ID tasku]_[názov dokumentu].

7. Priebeh stavu user story

Pre user story v TFS existujú nasledujúce stavy: **New, Approved, Commited, Done.**

1. Keď sa user story resp. backlog item vytvorí, automaticky je v stave **New**.
2. Po vytvorení a schválení špecifikácie a zadania taskov sa item presunie do stavu **Approved**, kedy môžu členovia začať na určených taskoch.
3. Keď sa dokončí posledný task, člen presunie user story do stavu **Commited**, v ktorom bude ponechaná až do konca šprintu, keď sa bude robiť review.
4. Ak sa user story podarilo dokončiť včas a spĺňa požiadavky, môže byť presunutá do stavu **Done**.

8. Priebeh stavu taskov

V TFS boarde máme vytvorené stĺpce na základe stavov, medzi ktorými je možné tasky presúvať: **To Do, In Progress, To Review, In Review, To Test, Testing, Done**. Medzi týmito stavmi sa budú tasky presúvať presne v momente, keď člen uvedie task do určeného stavu tj. začne na ňom pracovať, začne ho testovať apod.

1. Nové tasky sa je možné vytvoriť buď v backlogu na konkrétny item alebo na boarde v stĺpci **To Do**.
2. Keď priradený člen začne práce na tasku, presunie ho do stavu **In Progress**.
3. Po ukončení prác, ak si to task žiada, ho presunie člen do stavu **To Review**, a priradí ho osobe ktorá je na review určená.
4. Tá potom presunie task do stavu **In Review**.

5. Po dokončení review buď task v prípade implementácie funkcionality posunie do stavu **To Test** pre testovanie a priradí členovi na to určenému, alebo ak ide o vytvorenie dokumentácie, ponechá ho v stave a ak je potrebné, požiada kompetentného člena o zverejnenie na stránke projektu. Ak review vytvoreného kódu/dokumentu neprebehlo úspešne, člen vráti task do stavu **In Progress**, do opisu spíše body, ktoré je treba opraviť a priradí naspäť členovi, ktorý úlohu vykonával.
6. Po úspešnom testovaní alebo zverejnení výstupu tasku ho osoba, ktorá robila testovanie resp. zverejnenie je možné task presunúť do stavu **Done**.

9. Uzatváranie šprintu

Pri ukončení šprintu je potrebné urobiť sprint review a user story, ktoré boli dokončené uzavrieť a presunúť do stavu **Done**. Ak sa user story nepodarilo dokončiť, je vhodné ju rozdeliť pomocou tlačidla **Split**, ktoré otvorí okno a automatický vypíše nedokončené tasky. Po potvrdení vytvorí novú user story, ktorá sa presunie do nasledujúceho šprintu a znovu sa pre ňu odhadne effort, ktorý sa následne odpočíta od effortu na pôvodnej user story.

10. Záver

Na základe spísaných pravidiel by mal tím dosiahnuť čo najlepšie využitie nástroja TFS a takisto čo najpresnejšie riadenie a informovanie o stave vývoja. Tieto pravidlá sa tím snaží dodržiavať a konať inak iba v prípade dohodnutých výnimiek.

Príloha B – Metodika testovania

1. Základné informácie

Názov tasku: Metodika testovania

Číslo tasku: 9581

2. Úvod

Tento dokument obsahuje metodiku testovania, ktorá hovorí o tom, aké kroky majú byť splnené, aby Úloha - Task prešiel testom a mohol sa označiť ako Done.

3. Vypracovanie

Všetky úlohy, ktoré sú pripravené na testovanie by sa mali nachádzať v **To Test** stĺpci, kam ich presunie reviewer po tom, ako vykonal revíziu úlohy a usúdil, že implementácia/vypracovanie je správne. Pred testovaním úlohy tester presunie lístok do **In Testing** stĺpca. Tester je vopred určený na spoločnom stretnutí. V prípade, že tester nebol určený, hociktorý člen tímu si môže zobrať úlohu a vykonať testovanie.

Testovanie úloh:

11. Preštudovanie špecifikácie úlohy.
    a. Prípadné nejasnosti prejsť s osobou, ktorá danú úlohu programoval, prípadne so zadávateľom úlohy.
12. Určiť nástroj na testovanie, stratégiu a typ testovania.
13. Vytvoriť viaceré testovacie prípady.
    b. Testovacie prípady by mali zahŕňať aj okrajové prípady.
14. Stiahnuť zmeny v zdrojových kódoch, skompilovať.
15. Spustiť a vykonať testy.
16. Overiť, či všetko prešlo a správalo sa tak, ako malo.
17. Vytvoriť dokument o uskutočnenom teste, ktorý by mal obsahovať bližšie informácie o teste, testovacie scenáre, testovacie dáta, výsledky a záver.
18. Ak je všetko v poriadku presunúť úlohu do stĺpca Done, inak presunúť spať do In progress a oznámiť všetkým zainteresovaným na lístku, že test zlyhal.

4. Záver

Tento dokument opisuje kroky, ktoré je potrebné podniknúť, aby sa lístok mohol považovať za otestovaný.

Príloha C – Metodika commitovania kódu

1. Základné informácie

Názov tasku: Metodika písania a commitovania kódu

Číslo tasku: 9580

2. Úvod

Tento dokument obsahuje metodiku commitovania kódu, ktorá hovorí o tom, aké kroky a konvencie treba dodržiavať pri písaní commitovaní kódu do BitBucketu.

3. Vypracovanie

Repozitár projektu robocup na FIIT je BitBucket. Aby kód v repozitári ostal prehľadný a aby sa do repozitáru nedostával chybný kód, je treba dodržiavať isté konvencie a postupnosti.

3.1 Commitovanie

Každý jeden commit musí obsahovať commit správu. Formát commit správ by mal byť v tvare [+/-/R/F] #<id_ulohy>. „krátky opis činností“.

- ‘+’ - slúži pre pridanie funkcionality do kódu.
- ‘-’ - odstránenie alebo zakomentovanie funkcionality.
- ‘R’ - z anglického refactor. Pri zmenení kódu bez upravenia funkcionality.
- ‘F’ - z anglického fix. Opravenie funkcionality

ID úlohy je ID tasku v TFS.

Krátky opis činnosti by mala byť oznamovacia veta do 50 znakov. Správa by mala obsahovať informáciu o tom čo, prípadne prečo bolo niečo zmenené, nie však informáciu o tom ako to bolo zmenené (implementované).

3.2 Vetvy

Každá implementačná user story by mala mať vytvorenú vlastnú vetvu, na ktorej môže pracovať viacero ľudí. Meno vetvy by malo obsahovať názov tímu a názov user story. V master vetve ani vo vetve TP2018 by sa nikdy nemal nachádzať nefunkčný alebo upravovaný kód.

3.3 Merge

Merge je git príkaz na spájanie vetiev. Pri ukončení práce na vetve sa master vetva zlúči do aktuálnej pracovnej vetvy a následne sa vetva zlúči do Master. Takto ostane v master vetve vždy aktuálna a funkčná verzia.

Merge commit správa by mala mať formát:

Merge branch ‘<nazov_vetvy_z>’ into <nazov_vetvy_do>

4. Záver

Tento dokument opisuje konvencie a návyky, ktoré treba dodržiavať pri práci s repozitárom BitBucket.

Príloha D – Metodika písania kódu

1. Základné informácie

Názov tasku: Metodika písania a commitovania kódu

Číslo tasku: 9580

2. Úvod

Tento dokument obsahuje metodiku písania kódu, ktorá hovorí o tom, aké kroky a konvencie treba dodržiavať pri písaní kódu.

3. Vypracovanie

Kód projektu robocup na FIIT je napísaný v jazyku Java. Aby kód ostal čitateľný a do budúcnosti modifikovateľný, treba dodržiavať isté štandardné konvencie kódu v jazyku Java.

3.1 Komentáre

Komentáre v kóde sa používajú na vysvetlenie a objasnenie kódu. Komentárov v kóde by nemalo byť zbytočne veľa a mali by čitateľovi umožniť rýchlejšie a jednoduchšie pochopiť kód. Dva hlavné typy komentárov sú jednoriadkové a viacriadkové komentáre.

Jednoriadkové komentáre

- Jednoriadkové komentáre sa začínaju znakmi //
- Ich platnosť končí koncom riadku
- //jednoriadkovy komentar

Viacriadkové komentáre

- Viacriadkové komentáre začínajú znakmi /* a končia znakmi */
- Každý riadok (okrem prvého a posledného) by mal začínať znakom *
- Viacriadkový komentár môže začínať a končit na tom istom riadku
- ```
  /*
   * viacriadkový
   * komentár
   */
  ```
  ```

Komentáre by sa mali nachádzať v riadku pred kódom ktorý objasňujú (často v riadku pred názvom triedy, metódy alebo podobne). Veľmi krátke komentáre sa môžu však nachádzať aj v rovnakom riadku ako kód, ku ktorému je komentár naviazaný.

3.2 Deklarácie

Deklarácie by vo všeobecnosti mali byť krátke, ale výstižné. Je dôležité taktiež dodržiavať konvenciu veľkých a malých písmen. Konvencie sú nasledovné:

Triedy

- názov triedy by mal obsahovať podstatné meno
- názov triedy by mal začínať veľkým písmenom
- v názve triedy by sa mali používať celé slová, pričom každé ďalšie slovo začína veľkým písmenom
- príklad: public class Player, public class LowSkill

Rozhrania

- názov rozhrania by mal začínať veľkým písmenom
- v názve rozhrania by sa mali používať celé slová, pričom každé ďalšie slovo začína veľkým písmenom
- príklad: public interface ActionListener{}, public interface Runnable{}

Metódy

- názov metódy by malo byť sloveso alebo viacero slov, ktoré obsahujú sloveso.
- názov metódy by mal začínať malým písmenom
- v názve metódy by každé ďalšie slovo malo začínať veľkým písmenom
- príklad: run(), runFast()

Premenné

- názov premennej čo najlepšie vystihovať, čo sa v premennej bude nachádzať
- názov premennej by nemal byť rovnaký ako názov triedy v ktorej sa premenná nachádza
- názov premennej by mal začínať malým písmenom
- v názve premmenej by každé ďalšie slovo malo začínať veľkým písmenom

- jednoznakové premenné by sa mali používať len ako dočasné premenné
- príklad: i, myHeight, numberOfRecords

Konštanty

- názov konštanty by mal byť písaný veľkými písmenami (capslockom)
- v názve konštanty by medzi slovami mal byť podtrhovník ("_")
- príklad: int MAX_WIDTH = 5, int IMAGE_SIZE = 512

3.3 Formátovanie

Formátovanie kódu taktiež významne napomáha čitateľnosti kódu. Vo všeobecnosti je vhodné dodržiavať nasledovné konvencie:

- dĺžka riadku by nemala presahovať "veľkosť obrazovky". Vo všeobecnosti sa jedná o nie viac ako 80 znakov.
    - v prípade dlhších riadkov by mal byť riadok rozdelený po čiarke, prípadne pred operátorom
- každá premenná by mala byť v rámci možností definovaná vo vlastnom riadku
- každý príkaz by mal byť vo vlastnom riadku
- Bloky kódu v rámci riadiacej štruktúry (podmienky alebo cyklu) by mali vždy začínať a končiť zátvorkami { a }
- Kód v rámci bloku by mal byť vždy odsadený tabulátorom
- Príklad:

```
if(condition) {
        do something
}
```

4. Záver

Tento dokument opisuje konvencie a návyky, ktorými by sa mal programátor riadiť pri písaní kódu.

Príloha E – Metodika mergovania

1. Základné informácie

Názov tasku: Metodika mergovania kódu

Číslo tasku: 10004

2. Úvod

Tento dokument obsahuje metodiku mergovania vetiev - kódu, ktorá hovorí o tom, aké kroky a konvencie treba dodržiavať pri mergovaní kódu v BitBuckete.

3. Slovník

Merge – zlúčenie

Branch – vetva

Commit – odovzdať

Push – odoslať zmeny

4. Skratky

PR – Pull Request

5. Vypracovanie

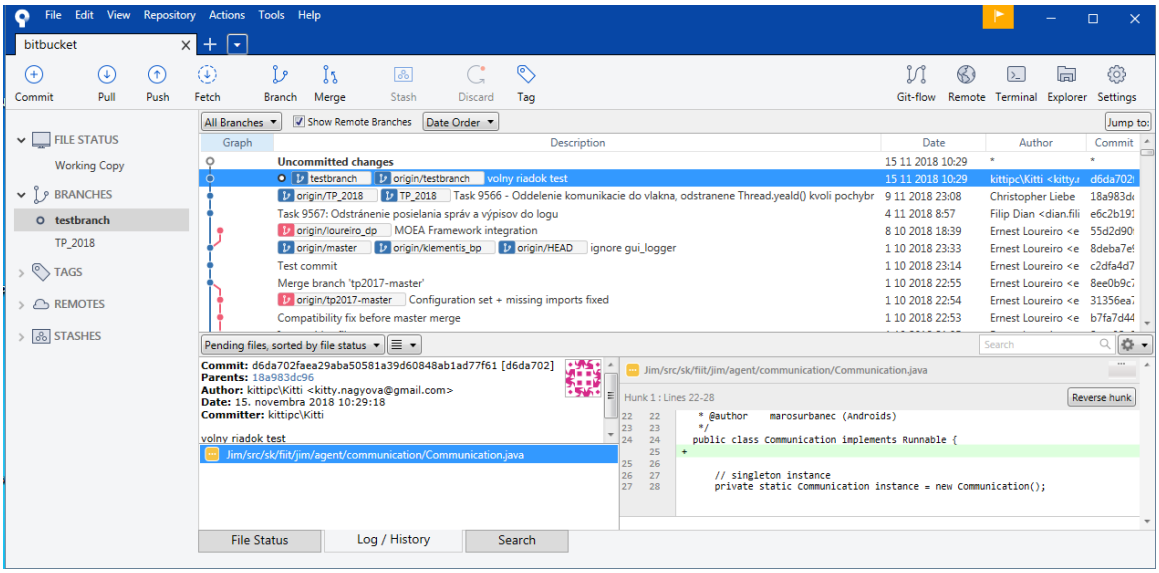
Projekt je uložený v repozitári na BitBuckete. V repozitári používame hlavnú vetvu Master a pracovnú vetvu TP_2018. Každá jedna user-story sa vyvíja zvlášť vo svojej vetve.

Pull Request

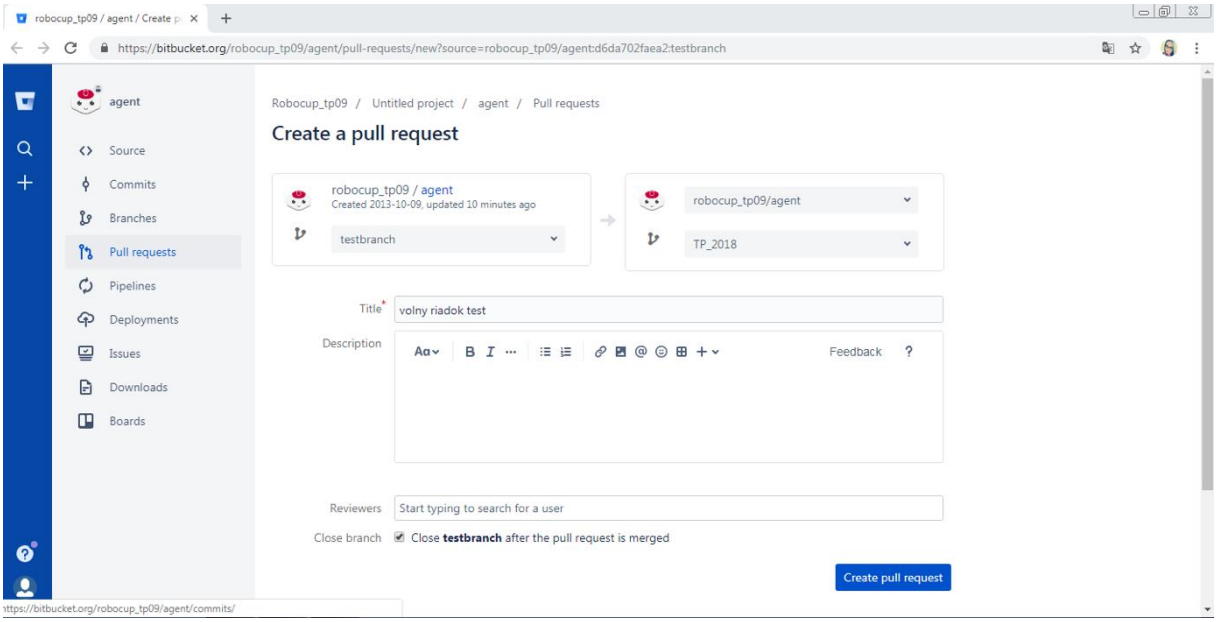
Pull request (PR) slúži na informovanie ostatných, že ste ukončili svoju robotu na vetve a daná vetva môže byť zlúčená do pracovnej verzie, resp. do mastra. Po tom, ako bol PR vytvorený, daná robota je pripravená na overenie (review) a následne pripravená na zlúčenie.

Kroky:

1. Pri začatí pracovania na user story sa vytvorí pre ňu nová vetva. Postup opísaný v metodike verziovania kódu.
2. Všetky zmeny vývojár commitne do vetvy pre konkrétnu user story. Nasledujúci obrázok zobrazuje novú vetvu „testbranch“ s testovacím commitom.



3. Po tom, ako vývojár ukončil svoju robotu na user story vytvorí PR. Pri PR si treba dať pozor z ktorej a do ktorej vetvy je vytvorený PR.



4. Reviewer si stiahne zmeny na danej vetve a vykoná review. Presný postup opísaný v metodike review kódu.

Mergovanie

Slúži na spojenie dvoch vývojárskych vetiev. Zlučovanie bude robiť Bc. Filip Dian. Vo výnimočných prípadoch môže iný člen tímu.

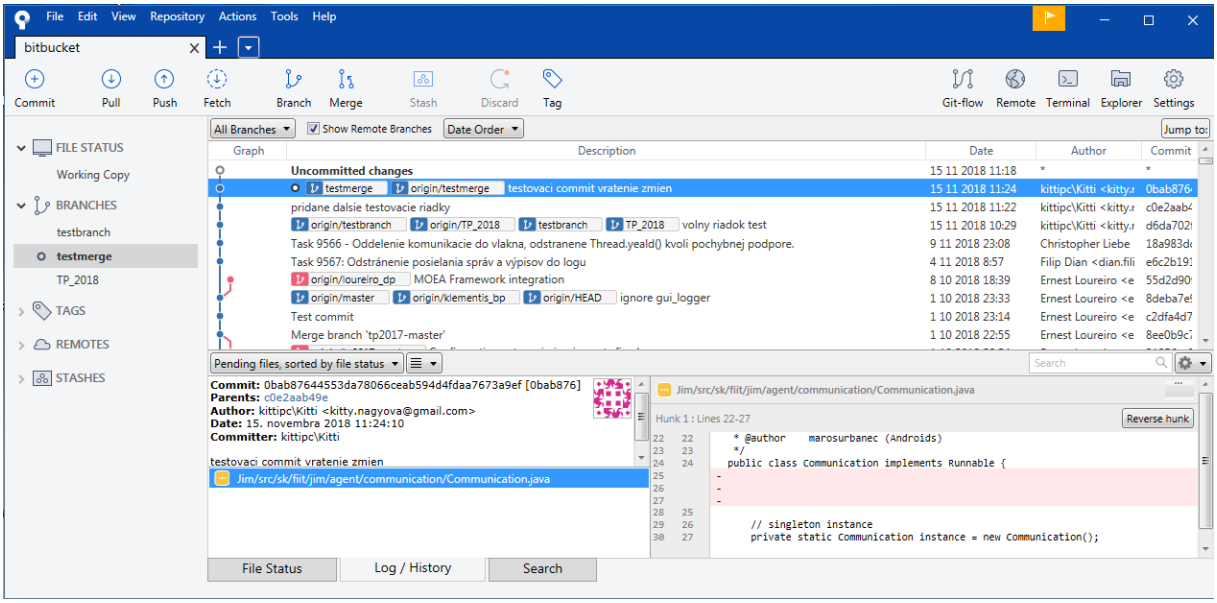
Kroky:

1. Po tom, čo bol vykonaný review na vetve a vetva je otestovaná, je možné ju zlúčiť s inou vetvou. User story vetvy sa najprv zlučujú s pracovnou vetvou *TP_2018*.

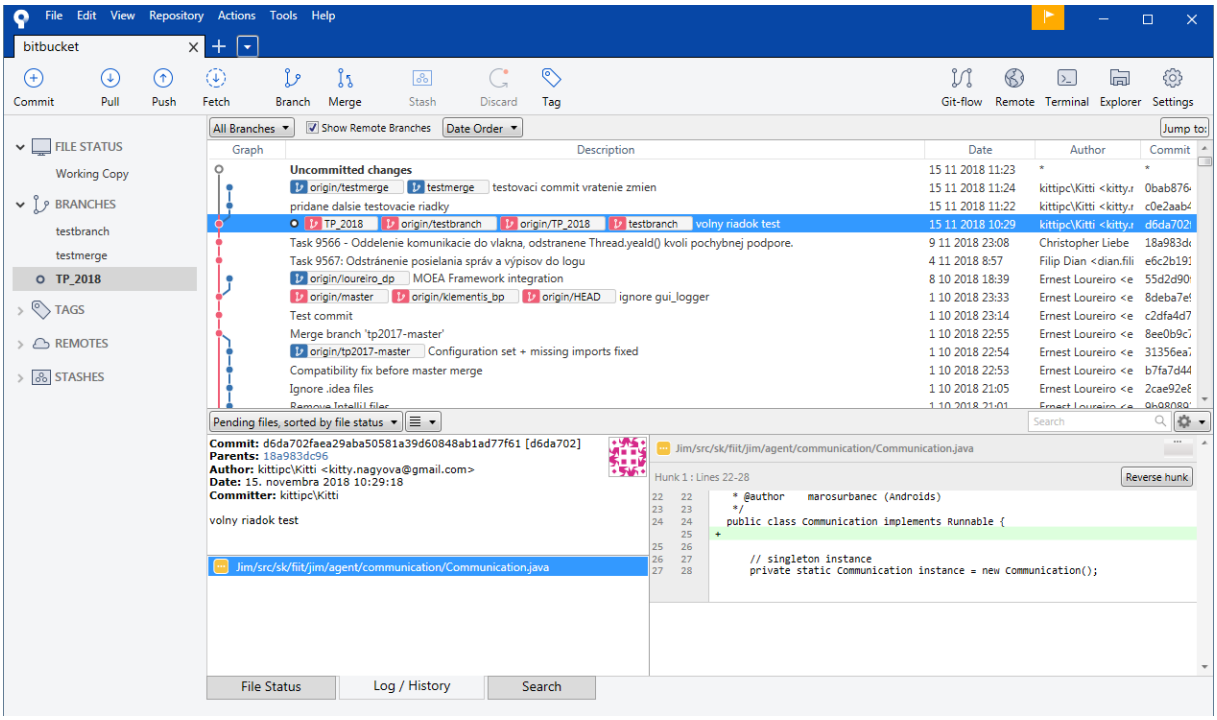
Do mastra sa bude zlučovať iba po dohode s nadriadeným.

2. Vykoná sa merge. Sourcetree ponúka mergovanie priamo v aplikácií. Postup:

    a. Na ďalšom obrázku vidíme, že vo vetve *testmerge* boli vykonané nejaké zmeny – commitnuté zmenené zdrojové kódy.



Ak si otvoríme aktuálnu verziu vetvy *TP_2018* vidíme, že tam vyššie uvedené zmeny nie sú.



b. Chceme teda pridať zmeny z *testmerge* vetvy do *TP_2018* vetvy. Najprv je potrebné prepnúť sa do hlavnej vetvy, do ktorej chceme mergnúť zmeny.

V Sourcetree to vykonáme dvojitým kliknutím na danú vetvu.

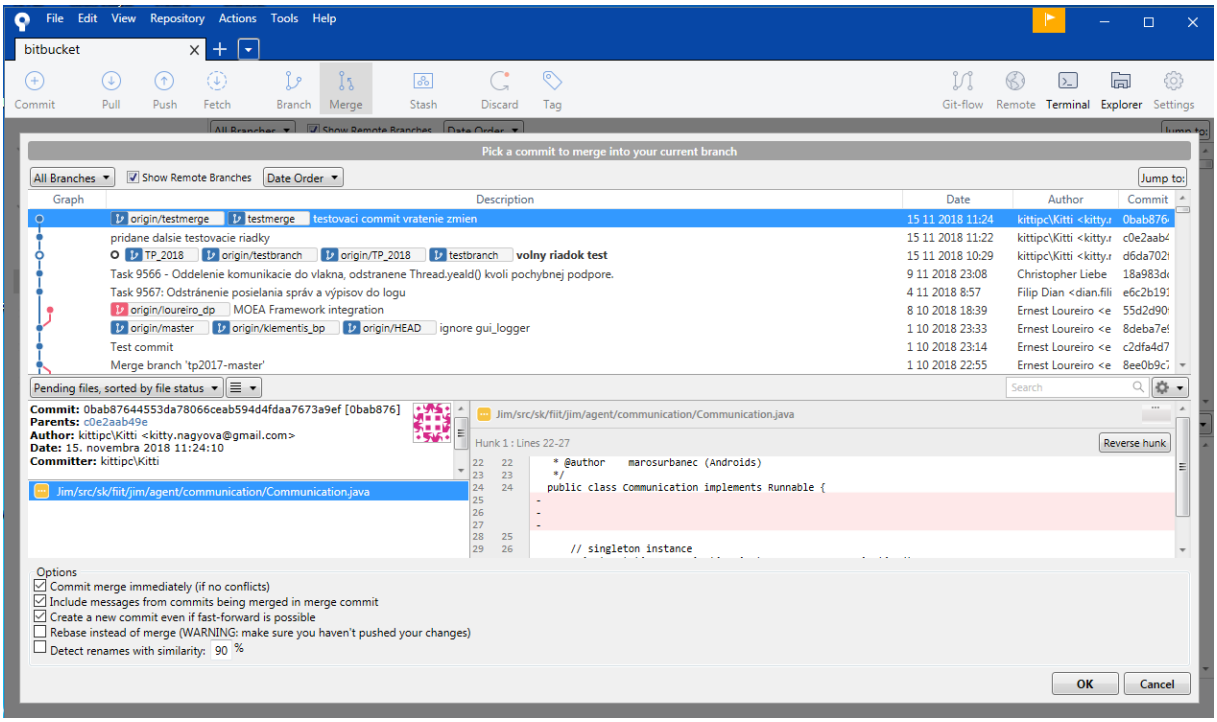
c. Klikneme na *Merge*. Vyberieme si posledný commit, ktorý chceme pridať do vetvy. Zaškrtneme checkboxy:

   i. Commit merge immediately (if no conflicts)

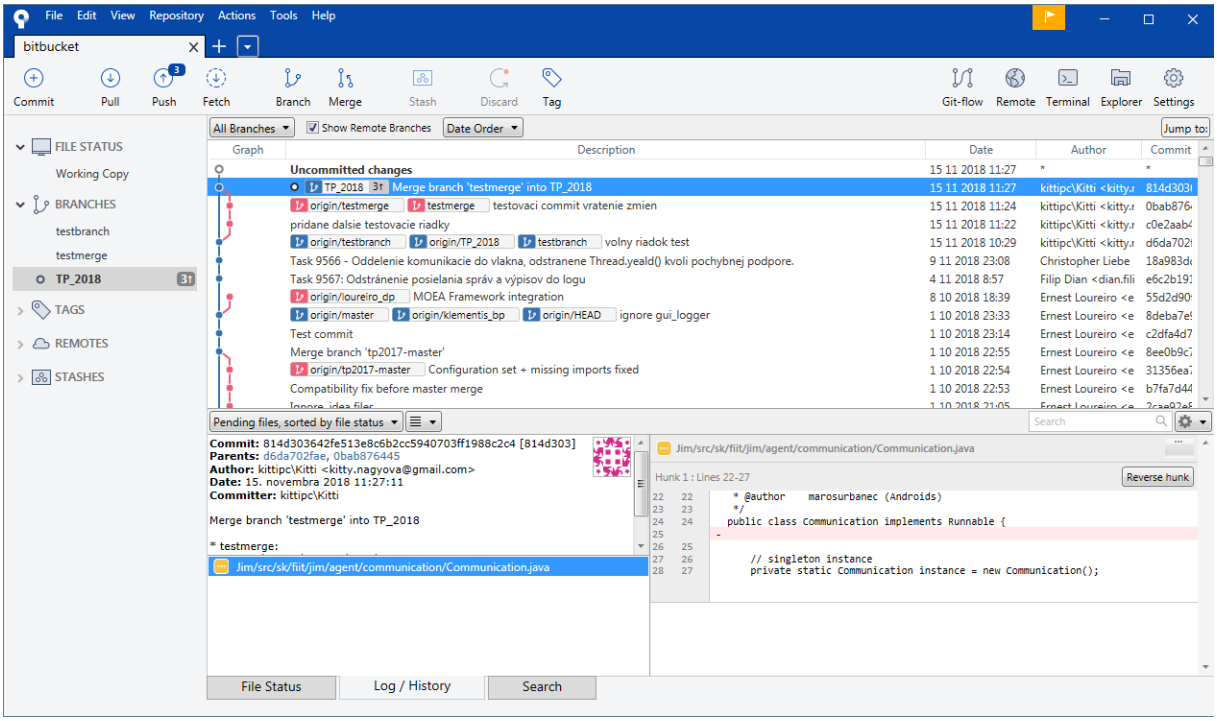
   ii. Include messages from commit being merged in merge commit

   iii. Create a new commit even if fast-forward is possible

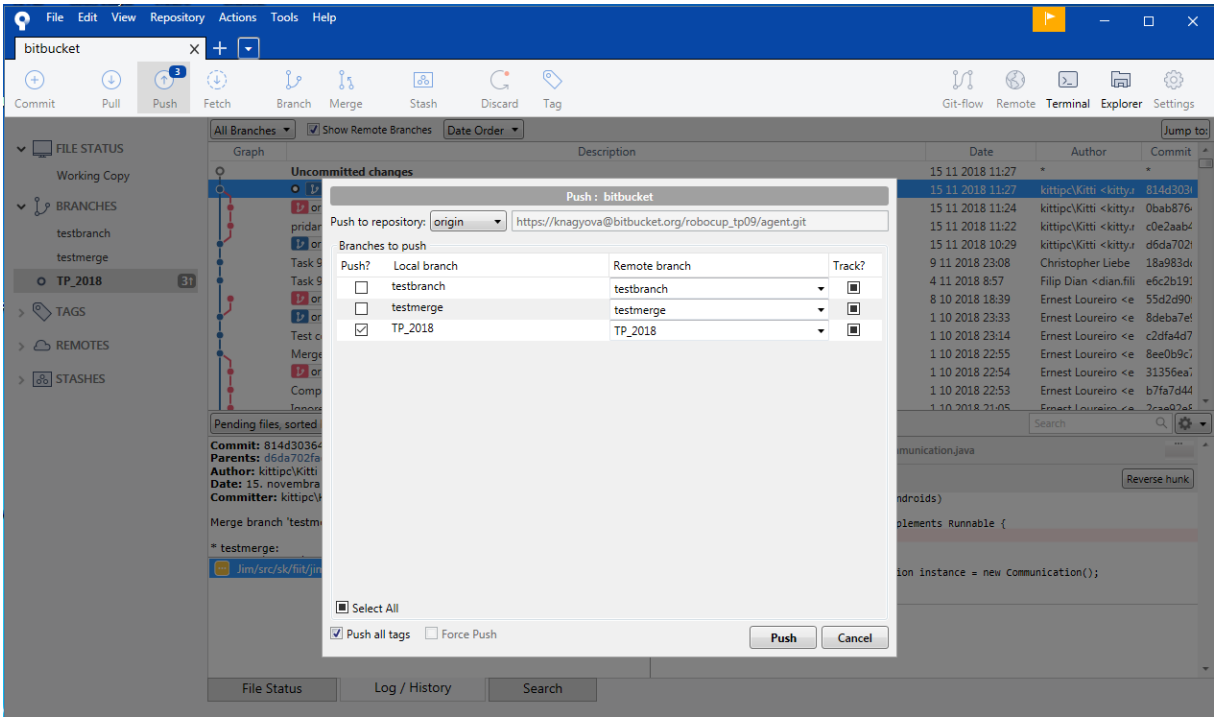
Klikneme na OK. Nasledujúci obrázok zobrazuje nastavenie Sourcetree pri mergnutí.



d. Vetvy sa zlúčili. Vidíme pridanú ikonku pri *TP_2018* vetve.



e. Nakoniec iba odošleme zmeny pomocou Push. Vyberieme si vetvu, ktorú chceme pushnúť.



f. Ak všetko prebehlo v poriadku, vidíme úspešne zlúčené vetvy a žiadnu chybu v Sourcetree.

6. Záver

Tento dokument opisuje postup a metodiku mergovania vetiev, ktoré treba dodržiavať pri práci s repozitárom BitBucket.

Príloha F – Metodika komunikácie

1. Základné informácie o úlohe

Názov úlohy: Metodika komunikácie

Číslo úlohy: 9583

2. Úvod

Tento dokument obsahuje opis metodiky komunikácie. V tomto dokumente nájdete opis jednotlivých spôsobov komunikácie v rámci tímu na danom projekte.

3. Vypracovanie

3.1 Verbálna komunikácia

Tento spôsob komunikácie sa praktizuje predovšetkým počas stretnutí (meetingov) v rámci tímu. Počas stretnutia komunikáciu vedie predovšetkým zadávateľ úlohy. Pre plynulý vývoj projektu a napredovanie sa počas stretnutia vedúci tímu musí resp. má za úlohu zisťovať potrebné informácie od členov tímu, ktorý by sa mali aktívne zapájať do diskusie. Takto získané informácie sú potrebné pre vyjasnenie nedorozumení a ku spresneniu požiadaviek na danú osobu. Komunikácia medzi členmi tímu má za úlohu aj ich korigovania v tíme a riešenia prípadných problémov. V prípade nedostačujúcej komunikácie počas stretnutia môžu dôjsť nedorozumenia v súvislosti s danými pridelenými úlohami, ktoré neskôr môžu vyústiť k nesplneniu úlohy resp. k nesúladnému vypracovaniu úlohy so zadaním, aký mal na mysli vedúci projektu/zadávateľ úlohy.

Standup

Stretnutia na komunikáciu a zamyslenie sa nad tým, čo sme urobili za posledných 24 hodín resp. za uplynulí čas od posledného stretnutia, čo plánujeme a čo urobíme v blízkej dobe. Tento spôsob komunikácie sa vedie postojačky a spravidla trvá krátko. Hlavným cieľom je osobné prediskutovanie vzniknutých problémov počas vývoja či splňania svojich úloh.

3.2 Elektronická – neverbálna komunikácia

E-mail

Vzniknuté problémy po spoločnom stretnutí je možné riešiť elektronickou formou s vedúcim projektu a to napísaním priameho e-mailu vedúcemu resp. na spoločným email, kedy o danom probléme budú informovaný všetci členovia tímu. Dôležité je pri písaní e-mailu uviesť do kópie

e-mailu aj emailové adresy zainteresovaných členov tímu do danej problematiky, aby boli súčasťou komunikácie a mohli sa vyjadriť k danej problematike.

Facebook

Tento spôsob komunikácie ja postačujúca v prvotnej fáze, v takzvanej fáze „rozbiehania projektu“ medzi členmi tímu. Tento spôsob komunikácie treba čím skôr nahradiť inou platformou/softvérom, ktorá je výlučne prispôsobená na komunikáciu pre tímy pracujúce na projektoch. Takýmto nástrojom je napríklad Slack, ktorému sa budeme nižšie v tomto dokumente venovať.

Nevýhodou je, že počas komunikácie na Facebooku nie je zainteresovaný aj vedúci projektu (zadávateľ úlohy). Táto forma komunikácie pri práci na rozsiahlom projekte je neprofesionálna a obmedzujúca z rôznych hľadísk. Jedným a hlavným obmedzujúcim faktorom je vyskakujúci multimediálny obsah na stránke prihlásenej osoby, ktorý výrazne rozptyľuje pozornosť.

Slack

Slack je cloudovo založený nástroj pre skupinovú komunikáciu a spoluprácu s tímom. Umožňuje vytvárať kanály (channels), ktoré obsahujú transparentné informácie z komunikácie medzi členmi tímu. Z toho vyplýva výhoda ľahkého vyhľadávania informácii z predošlej komunikácie. Členovia tímu môžu navzájom komunikovať v rámci kanálov alebo prostredníctvom súkromných správ (one-to-one komunikácia).

Slack taktiež podporuje integráciu nástrojov na riadenie projektov ako sú Asana, Jira, PivotalTracker, Blossom, TFS a Trello. Taktiež veľa aplikácii môžu byť prepojené so Slack-om, ako napríklad Google Drive, Dropbox, GitHub, MailChimp, Intercom, ZenDesk, Salesforce, Twitter a InVision.

Slack ďalej podporuje nastavenie pripomienky príkazom */remind* v kanáli. Pomocou tejto funkcionality budete v nastavenom čase upozornený o splnenie nejakej úlohy, ktorú ste si zadefinovali.

Umožňuje správu a sledovanie vášho zoznamu úloh, ktoré boli odkomunikované na stretnutí s tímom. Obrovskou výhodou je možnosť komunikácie formou video hovoru alebo klasickým zvukovým telefonátom s 15 osobami.

Kanály v našom tíme **Team#19**:

- random

- genereal
- stranka
- stretnutia
- tfs
- tasky
- odovzdavanievystupov

4. Nástroje na riadenie projektov

Nástroj Team Foundation Server (TFS) zachytáva podrobnosti o funkciách a kritériá akceptácie a umožňuje nám pokračovať v rozhovoroch o jednotlivých úlohách. Tento nástroj je sprístupnený všetkým členom tímu, ktorí sa na projekte podieľajú. Jeho správa by mala byť riadená administrátorom nástroja, ktorý má za povinnosť určiť práva jednotlivým členom tímu pre vykonávanie zmien v nástroji. TFS poskytujú celému tímu pohľad do projektu a spôsob, ako rýchlo posúdiť pokrok. TFS je priamo prepojený s komunikačným nástrojom Slack. Pri zmene stavu úlohy sa zobrazí notifikácia v Slack-u. Táto notifikácia obsahuje informácie o úlohe ako: číslo úlohy, názov úlohy, predošlí stav, nový stav, dôvod prechodu z predošlého stavu a dôvod prechodu do nového stavu. V tomto nástroji je ďalej implementované označenie členov tímu na lístku pre revíziu danej úlohy. Po revízii danej úlohy osoba, ktorá vykonala revíziu je povinná výsledok revízie zdokumentovať v rámci lístku formou pridania komentárov alebo otvorením diskusie k nezrovnalostiam v riešení danej úlohy. Táto forma komunikácie výrazne dopomáha k rýchlemu vyriešeniu danej úlohy.

5. Záver

Od každého člena tímu sa očakáva komunikatívnosť, pretože komunikácia medzi členmi tímu je hlavným aspektom agilného vývoja softvéru. V prípade akýchkoľvek problémov je potrebné daný problém v dostatočnom časovom predstihu odkomunikovať s príslušnou osobou pre zamedzenie vzniku ďalších chýb alebo obmedzovanie práce ďalších ľudí na danom projekte. Ďalším dôležitým aspektom v oblasti komunikácie je poskytovanie všetkých užitočných a dôležitých informácii všetkým členom tímu v priebehu práce na projekte a ich uchovávanie na voľne dostupnom mieste.

Príloha G – Definition of ready

1. Základné informácie

Názov tasku: Vytvorenie Definition of ready

Číslo tasku: 9564

2. Definition of Ready

Na to, aby sme mohli začať pracovať na novej User Story musíme vedieť, kedy je pripravená na riešenie (ready). Vo všeobecnosti by mali byť splnené kritériá I.N.V.E.S.T. [1]:

- **Independent**
  Úloha nie je závislá od žiadnych externých faktorov ani inej úlohy v rámci šprintu
- **Negotiable**
  Začleneniu US do šprintu predchádza krátka diskusia, počas ktorej sa vyriešia nejasnosti, v prípade potreby podniknú drobné úpravy a zjednotia ciele
- **Valuable**
  US by mala mať okamžite identifikovateľnú pridanú hodnotu pre produkt a koncového zákazníka
- **Estimable**
  Úloha musí byť časovo odhadnuteľná a splniteľná v rámci jedného šprintu
- **Small**
  Vypracovanie US by nemalo trvať dlhšie než polovicu šprintu
- **Testable**
  Úloha musí mať jasne definované akceptačné kritériá

Príloha H – Definition of done

1. Základné informácie

Číslo tasku: Task 9190

Názov tasku: Vytvorenie dokumentu Definition of done

2. Definition of done

Každý z bodov by mal byť splnený, ak je pre danú user story relevantný. User story sú v nástroji TFS reprezentované ako *Backlog* Item. Na základe typu user story sú definované kritéria, podľa ktorých sa hodnotí voči jej naplneniu. Hodnotenie prebieha na konci šprintu, kedy by zadané user story mali byť dokončené. User stories budú rozdelené na nasledujúce typy:

- Analýza
- Implementácia funkcionality

2.1 Definition of done pre user story typu „Implementácia funkcionality“

- Všetky tasky splnené
- Spracovaná dokumentácia k funkcionalite
- Spracovaná dokumentácia k použitiu v prípade funkcionality s externým ovládaním
- Dokončený kód
- Unit testy pre triedy s funkcionalitou
- Hotová revízia kódu
- Manuálne testovanie v prípade nutnosti otestovania scenára
- Splnenie akceptačných kritérií
- Funkcionalita nasadená
- Overená stabilita programu

2.2 Definition of done pre user story typu „Analýza“

- Všetky tasky splnené
- Pokrytie všetkých identifikovaných oblastí
- Návrhy na zlepšenie v konkrétnej oblasti
- Zhodnotenie a výsledky analýzy
- Spracovaná dokumentácia
- Splnenie akceptačných kritérií