Analýza videnia sveta, špecifikácia požiadaviek a návrh vylepšení

**Aká funkcionalita videnia sveta funguje**

**Natočenie agenta** - AgentRotationCalculator -> calculateRotation , ale je potrebne volat pravidelne updateRotation zo ziskanych dat zo servera. Data musia obsahovať staticke objekty, vlajky (minimalne 3 kusy).

**Pozícia agenta** – v agent.position v AgentModeli. Vypočíta to AgentPositionCalculator.updatePosition s údajov zo servera. Údaje musia obsahovať minimálne š videné vlajky ako statické objekty (s hodnotou absolutnej pozície).

**Pozicia dynamickeho objektu** – v triede DynamickObject je dokoncene sledovanie lopty, ale treba **dokončiť sledovanie hráčov**. Vieme zistiť poziciu dynamickeho objektu metódou getPosition(), vieme tiež získať relatívnu pozíciu vhľadom na čas videnia objektu. Je daný minimálny čas potrebný na zachytenie dynamického objektu v konštante LOW\_TIME. Je tam pokus o predikciu pozície na zaklade času kedy bol dynamický objekt naposledy videný a jeho rýchlosti. Ale nie je to dokončené. **Vylepšenie možné predikciou.**

**Údaje prostredia** – sú uchovávané v EnvironmentModel. Obsahuje údaje ako čas hry, čas simulácie, hrací mód (BEFORE\_KICK\_OFF, PLAY\_ON, KICK\_OFF\_LEFT, KICK\_OFF\_RIGHT, KICK\_IN\_LEFT, KICK\_IN\_RIGHT, CORNER\_KICK\_LEFT, CORNER\_KICK\_RIGHT, GOAL\_KICK\_LEFT, GOAL\_KICK\_RIGHT, OFFSIDE\_LEFT, OFFSIDE\_RIGHT, GAME\_OVER, GOAL\_LEFT, GOAL\_RIGHT, FREE\_KICK\_LEFT, FREE\_KICK\_RIGHT), časový krok, ktorým je čas zvyšovaný, stav hry (BEFORE\_KICK\_OFF, GAME\_OVER, GOAL\_LEFT, GOAL\_RIGHT, KICK\_OFF\_LEFT, KICK\_OFF\_RIGHT), verzia servera (podpora pre poslednu verziu pridaná 0\_6\_5)

pozicie statickych objektov – su pozicie vlajok. Pre každú verziu servera rozdielna implementácia. Tiez viem ziskat poziciu branok (našej a súpera).

**Taktické informácie o herných situáciách** – TacticalInfo

* isOffensiveSituation
* isDefSituation
* isBallOurs
* isBallTheir
* isStartAttackSituation
* isEnemyStartAttack
* positionInformation – ostatných
* pos
* relPos

zistenie ze lopta je najblizsie ku mne. Zistime ze je lopta najbližšie ku mne z mojho tímu. Existuje metoda ze ci nas tim zaklada utok a ci je pod tlakom (podla vzdialenosti hráčov od súperov), alebo za akú dobu ho založí. Tiež existuje metóda, ktorá zitťuje či súperov tím zakladá útok a či je pod tlakom. Máme aj metódu, ktorá zisťuje stav hry či útočíme alebo či bránime. Je nepriatel na nasej polovici a je naš hráč na našej polovici. Vieme zistiť počet našich obrancov obrancov. Je to ale dosť nepresné lebo to bere len na základe pozície našich hráčov na osi X oproti pozícii súperov. Vieme zistiť počet našich aj súperových útočníkov. Vieme zistiť či je lopta naša na základe vzdialenosti lopty od hráča (<1). Obdobne vieme zistiť či loptu má súper. **Formácie** – máme metódy, ktoré nastavujú pozície hráča vo formácii vzhľadom na pozíciu lopty. Je to implementované ako stavový automat. Určuje pozíciu pevne na ID hráča. Vylepšiť to vieme tak že budeme poziciu vo formácii určovať podľa blizkosti k bodom vo formácii. Ďalej brať do úvahy vzdielenosť od lopty (niekto zakričí, kto ide po lopte). Formácia je dynamická pri útoku a sekvenčná pri obrane, čiže musíme brať do úvahy čas a v čase meniacu sa pozíciu formácie.

**Prezentovanie agentov ako hráčov na ihrisku** – Player. Trieda s vlastnosťami hráčov. Vlastnosti vieme čítať a zapisovať. Vieme si vypočítať vzdialenosť lopty od bodu, lopty od hráča. Vzdialnosť nôh od seba a uhol. Tiež vieme zistiť či je agenyt v dosahu iného hráča. Vieme nastaviť chodidlá , ruky a hlav, vypočítať pozíciu uhol chodidiel vzhľadom na X os ihriska.

**Model sveta** –WorldModel dokoncit autogenerovanie logovacieho suboru vo fixtures. Drži a updatuje informácie o našich a superových hráčoch. Vie a zisťuje informáciu a pozícii lopty a robí aj **predikciu lopty na základe už zistených anotácii** (nie je to dokonalé). Vieme kontrolovať pozíciu hráča na dostrel do brány. Stavovy automat počíta na základe vzdialenosti hráča od súperovej brány. Určuje to pre nášho aj súperového hráča.

**Model agenta** – AgentModel – objekt triedy uchováva uhol kazdeho klbu hraca. Vieme aj získať pozíciu hráča (globálnu a relatívnu). Spracovať správu zo servera a podľa toho prisposobovať kĺby, natočenia,… v objekte. V agent modeli su aj matematicke funkcie na pracu s maticami a vektormi (opisujúcimi agenta). Vieme tiež získať rýchlosť a zrýchlenie hráča. Vieme zistiť či agent stojí alebo je na zemi, či agent leží na bruchu alebo na chrbte. Vieme zistiť kedy videl agent naposledy vlajku. Pomocou anotácii vieme upraviť objekt sučasného agenta.

**Aku treba vytvorit alebo co treba vylepsit**

**Odhadové videnia sveta** – šanca získať loptu našim hráčom = dotknut sa jej

* kto rychlejsie kopne (som chrbtom, na zemi) – posielanie sprav s casmi kopov, odhadnem vzdialenost a svoju rychlost = to iste ako predchadzajuce, len špec. prípad
* som brankar, obranca, stred, utocnik = urcovanie pozicii
* prekazka? z matematickeho hladiska
* diery medzi prekazkami k superovej brane ak mam loptu
* ak mam loptu a diery medzi prekazkami kde je najblizsi hráč = mozem prihrat
* bezpecny bod na priamke supera k branke = kde najlepsie branim

**Čo treba dokončiť**

* sledovanie hráčov a držanie si informácie v dynamickom objekte je potrebné dokončiť
* možné vylepšenie je pomocou predikcie zistiť poziciu lopty
* integráciu parsera správ servera do modelov agenta (jeho videní)

**Rozdelenie údajov sveta**

**Dynamické objekty** – možné vylepšenie dokončením predikcie pozície lopty na základe získaných anotácii alebo rozšírením zachytávania do dynamického objektu aj hráčov. Lopta a hráč sa v údajoch nelíšia v pozícii, zrýchlení, rýchlosti a času posledné videnia, ale hráč by mohol mať navyše údaje ako id a team.

**Statické objekty** – pozície vlajok alebo bránok.

**Agent** – objekt obsahuje informácie o mne. Moje kĺby, pozícia (absolútna alebo relatívna), moja poloha, rýchlosť, zrýchlenie

**Hráč** – objekt reprezentujúci hráčov na ihrisku a teda aj mňa. Obsahuje informácie potrebné iba o hráčoch a to je ich pozícia, uhly nôh, rúk a hlavy a či je hráč na dosah iného hráča alebo vzdialenosť lopty od hráča. Je to prezentácia dynamického objektu hráča, ale chýba zrýchlenie, rýchlosť a možnosť predikcie.

**Prostredie** – objekt reprezentujúci stav hry, mód hry, čas hry, informácie o hráčoch na ihrisku a o lopte a to pozície. Nedokončená predikcia pozícia hráčov a lopty.

**Taktické informácie** – objekt reprezentujúci informácie o stave tímu či útočí alebo obraňuje, aká je formácia a kde je moje miesto vo formácii. Sú to dôležité informácie pre vyššie pohyby. Obsahuje tiež informácie o počte útočníkov a obrancov na oboch stranách ihriska alebo či je lopta naša. Formácia sa udáva vzhľadom na pozíciu lopty či je na našom alebo na súperovom ihrisku a či loptu vlastníme my alebo súper.

**Návrh vylepšení**

**Vylepšenie taktiky pridelovaním najbližšej pozície vo formácii**

Pozícia vo formácii sa hráčovi prideľuje na základe jeho id. To môže byť nevhodné pokiaľ hráč bude z nejakých dôvodov na inom konci ihriska a daná pozícia môže byť obsadená bližším hráčom skôr.

Preto sme navrhli vylepšiť algoritmus počítaním si vzdialeností k jednotlivým pozíciam, výberom najbližšej voľnej pozície, zahlásením obsadenej pozície a príchodom do pozície.

Nasledovný pseudo algorirtmus využíva už existujúcu funkcionalitu. Základná časť algoritmu je nasledujúca:

*foreach formPosition in positions*

*dist[][0] = formPosition.vector – my.position.vector;*

*dist[][1] = formPosition.id*

*end*

*dist.sort();*

*have\_formPosition = falsel;*

*while ( !have\_formPostion)*

*myPosition = dist[0];*

*if ( !positioHold(dist[0][0]) )*

*Communication.transmit(„(say Position: „ + dist[0][1] + “ taken)“);*

*have\_formPosition = true;*

*else*

*dist[0].pop();*

*end*

*end*

*positionHold( positionId )*

*foreach position in holdPositions*

*if ( positionId == position.id)*

*return true;*

*else*

*return false;*

*end*

*end*

*end*

**Šanca získať loptu našim hráčom**

Keď loptu nevlastní žiaden hráč, je dôležité si určiť kto loptu získa skôr. Či to bude náš tím alebo súperov. Preto sme navrhli algoritmus kde hráč zistuje poziciu hráča najbližšieho k lopte vrátane seba. Počíta to z pozícii hráčov ktorých vidí a z pozície lopty, ktorú vidí. Táto informácia pomôže hráčovi sa následne rozhodnúť či isť po loptu alebo sa pripraviť na bránenie súperovmu hráčovi. Algoritmus berie do úvahy aj či je hráč otočený chrbtom k lopte. Štruktúra algorimtu je popísaná nasledovným pseudokódom:

*ballPos = WorldModel.getInstance().getBall().getRealtivePosition();*

*min = infinity;*

*foreach player in players*

*if ( player is turned ) player.relPosition += turnDdistance;*

*if ( palyer.relPosition – ballPos < min )*

*min = player.relPosition;*

*playerId = player.id;*

*playerTeam = player.team;*

*end*

*if ( playerTeam == ours ) return 100%;*

*else return 0%;*

*end*