Virtuálni agenti
v zložitom virtuálnom prostredí

1. Ciele práce

Cieľom je vytvoriť dvojicu agentov (botov) v zložitom virtuálnom prostredí FPS hry Unreal Tournament 2004 (UT2004). Ďalej vyriešiť ich spoluprácu v mode DM (DeathMatch). Projekt je vypracovaný v rámci a pre predmet FI:PA026.

1.1 Výstupy

Výstupom je architektúra (viz. 3.3) a dva prototypy spájajúce existujúce technológie (Pogamut, POSH plán, BOD[1], UT2004):

- **DMTeamBots** – Riešenie obsahujúce logiku a chovanie zabalenej do jednej triedy *Behaviour.java.*

- **YaPOSH-DM –** Modulárny prístup k tvoreniu zmyslových a akčných primitív. Funkčná parametrizácia v POSH plánoch.

Obe riešenia spájajú v sebe funkčnú logiku s dvoma jemne odlišnými reaktívnymi POSH plánmi (viz. 3.3). Takýto boti sú schopný spolupracovať posielaním si správ cez rozhranie hry, reagovať na hrozby, plynulo sa pohybovať po mape. Úspešnosť riešenia je testovaná v sekcií 4 – testy.

2. Inštalácia

Inštaláciu je možné rozdeliť do niekoľko fázy:

1. Inštalovanie hry UT2004, tu je možné najjednoduchšie zakúpiť cez klienta Steam (http://store.steampowered.com) od firmy Valve Software.
2. Nainštalovanie a nastavenie JDK 1.6 – 1.8 a a IDE NetBeans verzia 7.3.+ pre správne fungovanie pluginov (UT2004 Servers, yaPOSH, Shed, Dash).
3. Inštalácia Maven-u
4. Inštalácia Pogamut 3.7.0 (http://pogamut.cuni.cz)

Video s tutoriálom na inštalovanie celého súboru programov okrem UT2004 je možné nájisť tu[2].

1 Behavioral Oriented Design – Je metodológia na tvorenie komplexných virtuálných agentov, ako napríklad virtálne charaktery z hier.

2 Autor tutorialu Nil Roessler z Lehigh University.

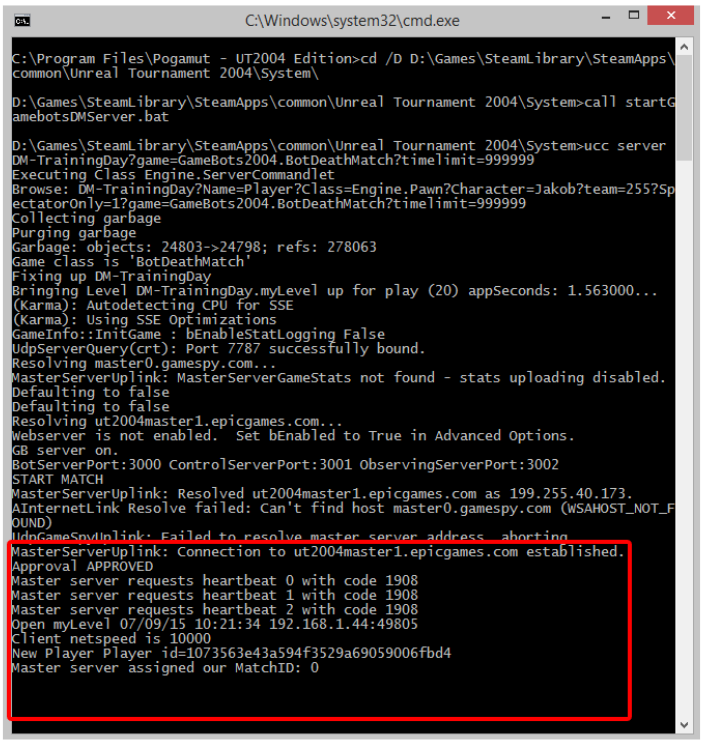
2.1 Setup pre botov bez Nativných protivníkov

Pred prvým spustením jar filu DM_BOTS je potrebne aby bežal server so zvoleným modom hry. Batch files je možne nájisť v priečinku úspešne nainštalovaného Pogamut-u (pre windows 7 – 8.1 je relatívna cesta Program Files\Pogamut - UT2004 Edition).

Typy batch filov:

- startGamebotsCTFServer – spúšťa server pre CTF(Capture the flag) mód hry
- startGamebotsDMServer – spúšťa server pre DM mód hry

V našom prípade je použijeme startGamebotsDMServer. Server beží pokiaľ je mu pridelený MatchID:



Obrázok 1: Riadne bežiaci server. Pridelene MatchID: 0

Pre overenie správania botov je potrebné sa do hry pripojiť ako hráč / pozorovateľ cez počítačovú hru UT 2004. Hra má obmedzený počet hráčov na 16.

Obrázok 2: Pripojenie sa na server v hre. JOIN GAME (1) -> LAN (Tab hore) a JOIN (2) -> Ako hráč (3) alebo ako pozorovateľ (4).

2.2 Setup pre botov s Nativ Bots

Postup je rovnaký ako pri 2.1 Setup pre botov bez Natívnych protivníkov. Natívne AI pridáme cez NetBeans plugin UT2004 Servers → Initialize server name(Right Click) → Connect Native Bots → BotType je úroveň inteligencie bota, hodnoty sú 1[Easy] – 3[Unfair].

Plugin umožňuje meniť mapy.

Video tutoriál so spustením je nájdete tu[3].

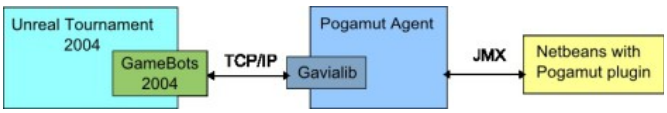
3 Autor Daniel Bendík

3 Implementácia

3.1 – Použité technológie

Ako zvolené virtuálne prostredie poslúžila hra Unreal trounament 2004 prípadne UDK (Unreal engine 3.0 +). Ide o FPS (First-person Shooter) akčnú hru zasadenú v realistickom 3D prostredí s rozličnými mapami.

Pre ovládanie, navrhovanie a programovanie je použitý framework Pogamut UT2004 + JAVA vyvíjaný na Fakulte Informatky Karlovej Univerizte v Prahe.
Agent momentálne používa framework Pogamut 3.7.0. Ten zapúzdruje sieťový textový protokol pre spojenie do UT2004.



Obrázok 3: Pogamut architektúra

Pogamut implementuje reaktívne plánovače ako pyPOSH, SPOSH, yaPOSH + Shed. (S)POSH je reaktívny plánovač vyvinutý na katedre informatiky na Univerizte v Bathu. Spája v sebe metodiku behaviorálneho návrhu logiky agentov. Implementuje strom chovaní používajúci prioritne selektory. Skladá sa elementov:

1. Drive Collection (DC) – Ide o Root stromu (nenáchadza sa vo vizulanom zobrazení pyPOSH plánu cez Shed)
2. Primitiv Action/Sense – Listy stromu a triggre
3. Competence (C) – Nižšia vrstva DC
4. Action Patterns – Zoznam akcii vykonávaných sekvenčne

Aktuálne agent používa na riadenie logiky yaPOSH (yet another POSH). Ten je zatiaľ najstabilnejšou verziou (S)POSH v JAVE.

3.2 – Štruktúra projektu

Štruktúra projektu pozostáva z 7 balíčkov.

- Zložené zmysly a akcie
  - **Shooting** – Modul zodpovedný za výber a správu zbraní. Implementuje rozhrania definovane v Pogamut API určené pre kontrolu zbraní a mierenia. Preferovanie silnejších zbraní.
  - **Utils** – Matematické funkcie a textový parser.
- Komunikačný modul
  - **Communicator** - Modul pre komunikáciu medzi botmi. Skladá sa z triedy definujúcej formát správy *Command* a triedy *Communicator* obsluhujúcej

posielanie správ cez Global Chat UT 2004.

- KB modul
  - **Memory** – Tento modul ma triedu *Memory*, ktorá predstavuje krátkodobú pamäť bota. V nej bot uchováva informácie o poslednom ciele / zbieraného predmetu, botov aktuálny stav a stav tímového spoluhráča. Pre prístup k základným informáciám o stave sveta využíva interface IUT2004.
- Riadiace Moduly
  - **Bot**
    - *Context* – Trieda poskytujúca prístup k objektom a metódam, ktoré sprístupňujú súčastný stav agenta (bota). Napríklad k viditelným predmetom, ako aj akutátorom pre ovplyvňovanie tohoto stavu. V **DMTeamBots** túto triedu predstavuje *Behavior.java*.
    - *Logic* – trieda zodpovedná za exekúciu logiky bota, obsahuje tiež exekútor yaPOSH plánu.
- Primitíva pre yaPOSH
  - **Action –** Základne akcie použite v yaPOSH plane.
  - **Sense –** Základne parametrizované zmysly a triggre v yaPOSH plane.

Primitíva yaPOSH plánu musia byť v pred-definovaných balíčkoch a to z dôvodu implementácie yaPOSH a správnym fungovaním anotácie pre parametrizáciu.

3.2.1 – Lokalizácia modulov

Relatívne cesty umiestnenia súborov / balíčkov pre **DMTeamBots**:

1. Moduly Shooting, Utils, Communicator, Memory**:**

   FIPA026\Bots\DMTeamBots\src\main\java

2. Logika, Chovanie:

   FIPA026\Bots\DMTeamBots\src\main\java\cz\cuni\DMTeamBots

3. SPOSH plán:

   FIPA026\Bots\DMTeamBots\src\main\resources\sposh\plan\BotPlan.lap

Umiestnenie pre **yaPOSH-DM**:

1. Moduly Shooting, Utils, Communicator, Memory:

   FIPA026\Bots\yaPOSH-DM\src\main\java

2. Logika, Kontext:

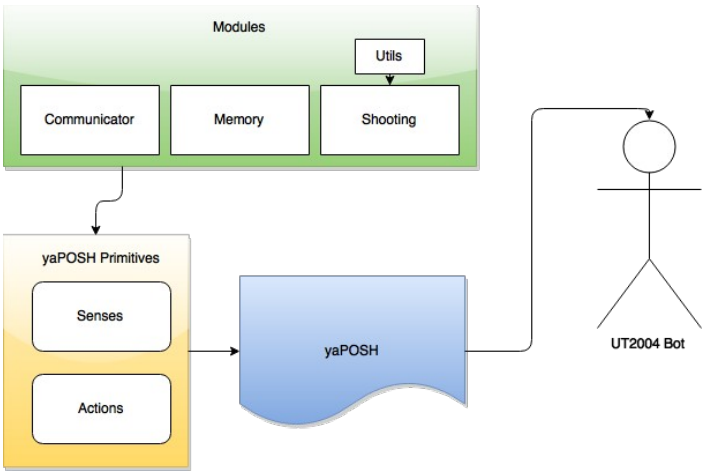
   FIPA026\Bots\yaPOSH-DM\src\main\java\cz\cuni\amis\pogamut\ut2004\examples\yaPOSH

3. yaPOSH plán:

FIPA026\Bots\yaPOSH-DM\src\main\resources\sposh\plan.lap

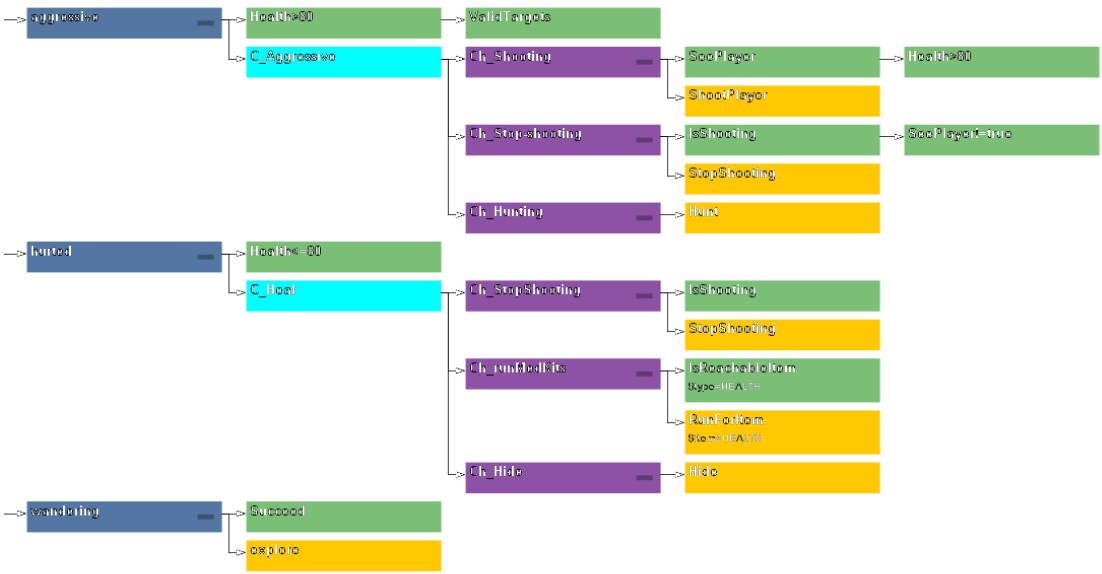
3.3 – Architektúra

Logická schéma agenta prepojenia modulov a tried je znázornená na obrázku 2.



Obrázok 4: Logická schéma

Zeleným sú označené moduly riadiace komunikáciu, pohyb a mierenie. Žltým sú jednotlivé primitíva využívané v yaPOSH a viditeľné v yaPOSH pláne obrazok 4. Primitíva pre zložitejšie akcie a zmysly využívajú vyššie spomínané moduly. Tento plán následne riadi rozhodovanie botov.



Obrázok 5: pyPOSH plán – Driver(modra), Kompetencia(azurova), Voľby(Fialova), Zmysly/Akcie(Zelena/Žlta)

Na yaPOSH pláne je vidieť 3 druhy driverov, ktoré predstavujú v logike agenta

„trichotómne“ chovanie. A to z dôvodu aby sa zabezpečilo, že agent sa nezasekne.

Agresívne chovanie je podmienené zdravím bota(v danom plane sa používa fixná hodnota no v reále je parametrizovateľná a ovládaná dynamicky) a zloženým trigerom *seeValidTargets*. Ten určuje či existujú v blízkom viditeľnom priestore nepriatelia. Pokiaľ sa v ňom nachádza nepriateľ je označený za validný cieľ, ale len počas určitého časového obdobia. Validné obdobie je variabilné, tzn. čas ktorý je nepriateľ viditeľný + fixná doba hľadania pokiaľ nepriateľ ešte existuje no nieje viditeľný. Chovanie presnejšie definuje kompetencia s 3 možnosťami: prvé dve zabezpečujú správne strieľanie na hráča + overenie či nejde o spoluhráča.
Ak prvé dve zlyhajú nastáva sledovanie poslednej stopy výskytu nepriateľa a pokus o prenasledovanie.

Hurted (Defenzívne) chovanie sa spustí ak je bot príliš poškodený. Pre správny chod a šetrenie munície sa overuje v kompetencii aj podmienka či sa ešte vykonáva akcia strielanie. Následne bot hľadá v priestore najvýhodnejšie zdroje života (“lekárničky”), pokiaľ nemá v dosahu žiadne snaží sa pred útočnikom ukryť.

Wandering (Túlanie) chovanie ak všetko ostatné zlyhá agent behá po mape a zbiera hodnotné predmety (munícia, zbroj, extra-zdravie).

3.4 Príklady chovania

Táto kapitola obsahuje ukážky chovania na schéme mapy DM-TrainingDay a ich videoukážky[4] priamo z prostredia UT2004.

Vysvetlivky elementov mapy:

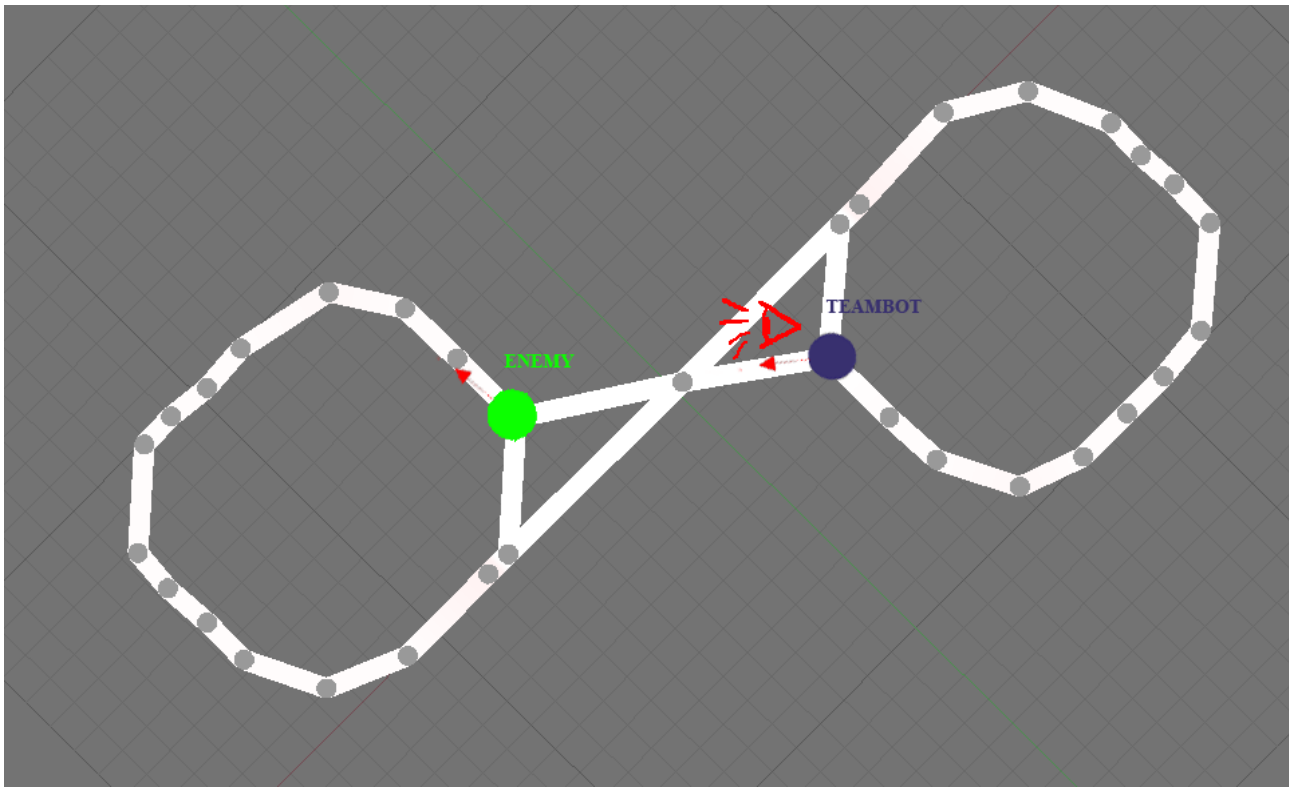
- Šedé body – navigačné body(ďalej len NB)
- Biele koridory – hrany medzi NB
- Farebné kruhy s červenou šípkou – Virtuálny agenti (Boti), červená šipka označuje orientáciu bota.
- Červené číslovanie – poriadie prechádzania NB.

Vysvetlivky k videam:

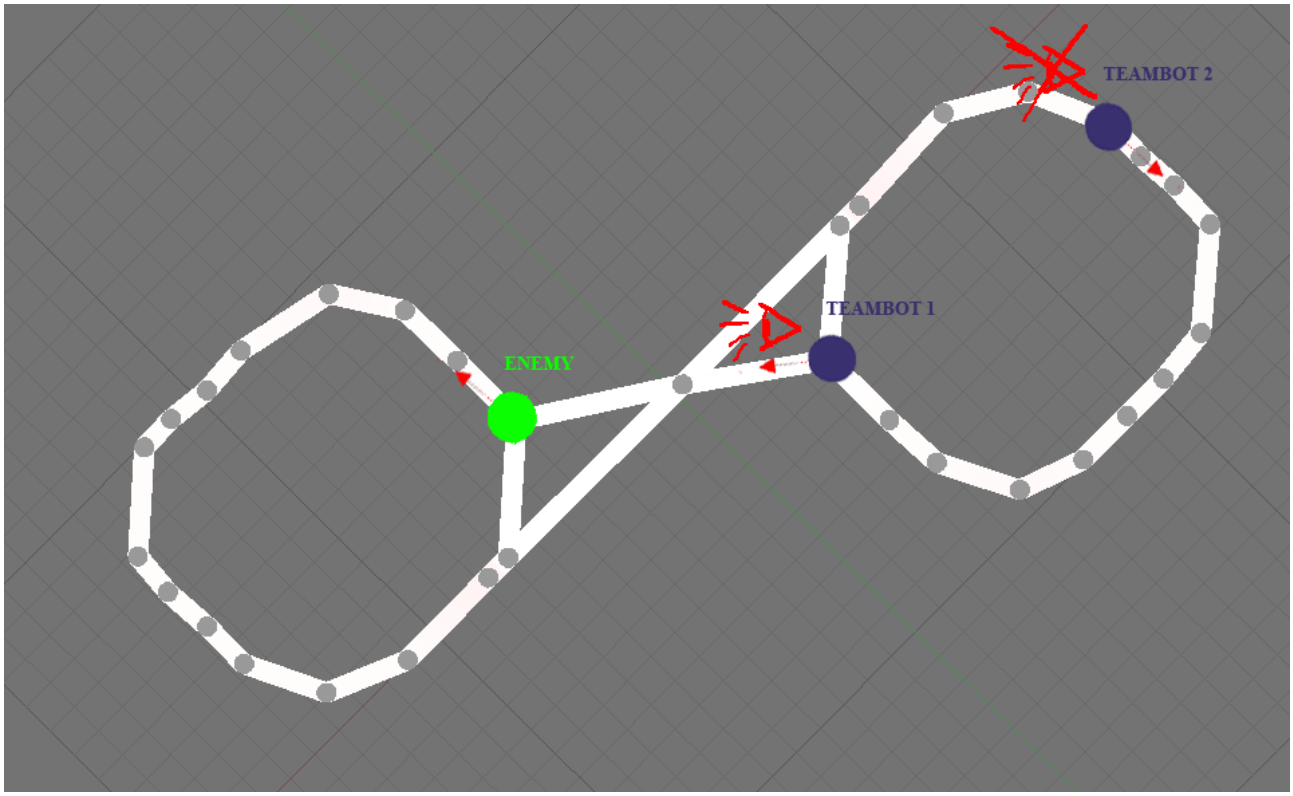
- Zmeny stavov – V ľavom dolnom rohu, žlté kapitálky (WANDER, HURTED, AGGRESSIVE, ATTACK [x,y,z])
- Status bota – Vertikálne stĺpce v podobe spritu prilepeného na pozíciu bota
  - Červený – život
  - Žltý – zbroj
  - Modrý – munícia

4 Autorom všetkých videií v sekcíí 3.4 je Daniel Bendík

Agresívne chovanie jedného bota:



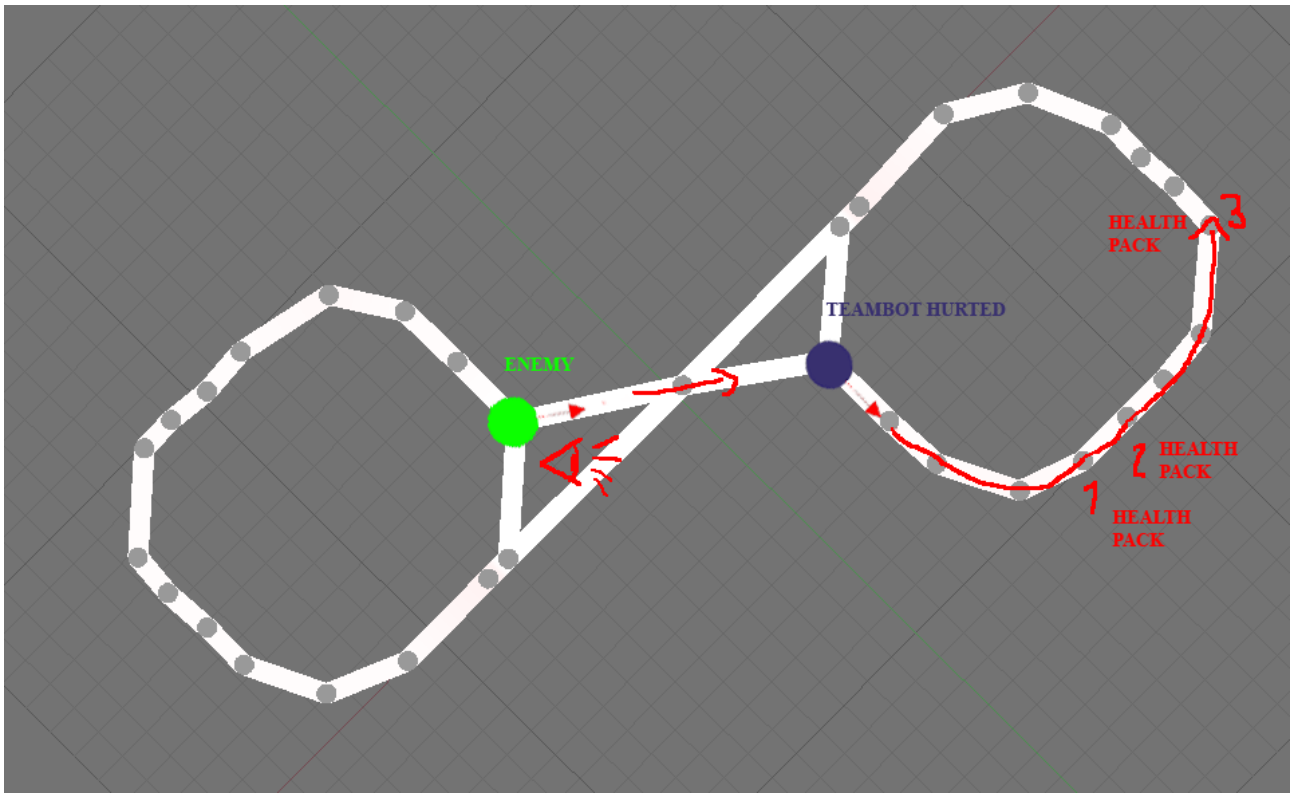
Obrázok 6: TeamBot (fialová) vidí nepriateľa (zelena) - prechádza do Agresívneho chovania.



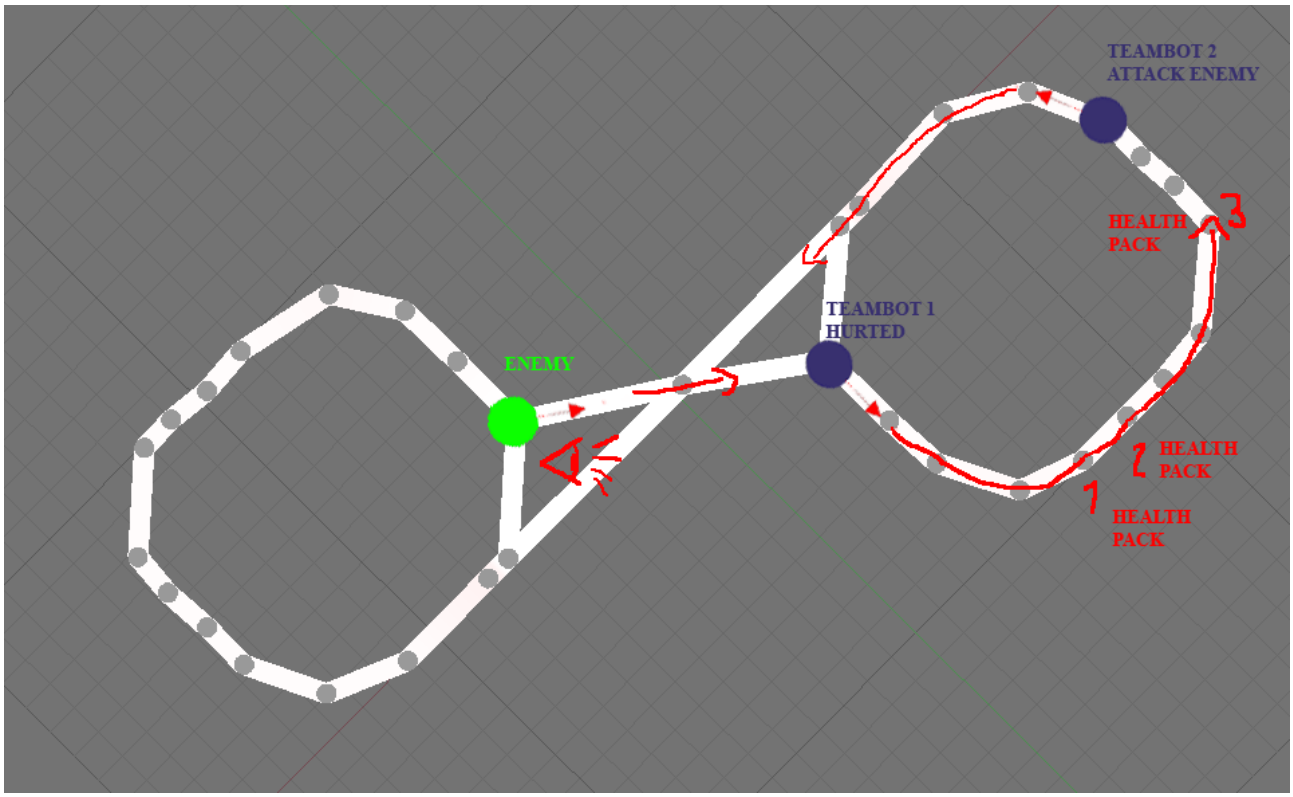
Obrázok 7: TeamBot 1(fialová) vidi nepriateľa pošle správu TeamBot 2(fialová, hore), ktorý nemá vizuálny kontakt, o pozícii nepriateľa.

Video s príkladom agresívneho chovania nájdete tu.

Hurted(defenzívne chovanie):



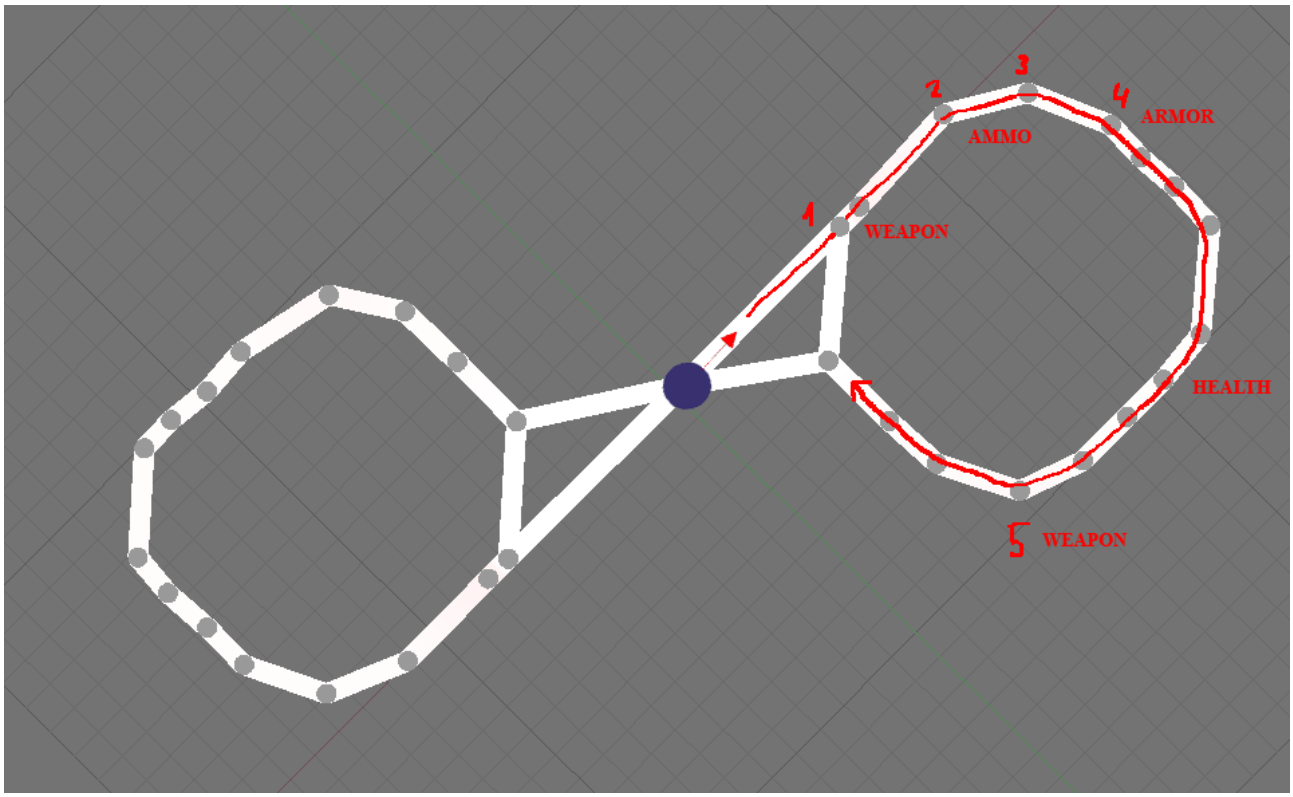
Obrázok 8: Nepriateľ(zelená) poškodí TeamBota(fialová). TeamBot prechádza do Hurted chovania. Prechádza najbližšie navigačné body(šedá) obsahujúce lekárničky(1,2,3).



Obrázok 9: Obdobná situácia ako na obrázku 8. Okrem prechodu na Hurted chovanie bot informuje svojho spoluhráča o pozícií nepriateľa čo na neho útočí.

Video s príkladom defenzívneho(Hurted) chovania nájdete tu.

Wandering chovanie:



Obrázok 10: Bot prechadza jednotlivé NB a zbiera potrebné predmety (munícia, zbrane, extra životy, zbroj).

Video s príkladom Wandering chovania nájdete tu.

Všetky chovania pokope je možne vidieť na tomto odkaze.

4 Testy

Testovanie pozostávalo zo súperenia pyPOSH DM botov proti Natívnym botom z hry UT2004 na najvyššej obtiažnosti. Scenár je rozdelený na 4 kategórie:

- **1 vs 1** – Jeden na jedného. Snaha o overenie navrhnutého dizajnu pyPOSH plánu a jeho modulov v boji proti natívnemu botovi.
- **2 vs 2** – Overenie komunikácie pyPOSH botov. A výhody takéhoto riešenia oproti neorganizovaným botom.
- **2 vs 1 & 1 vs 2** – v dvoch variantach 2 komunikujúci pyPOSH boti proti jednému protivníkovi a naopak.

V každej z nich sa odohralo 10 hier na mape Albatross – ako jedna z mála so spojitým navigačným grafom a zároveň dostatočne rozsiahla a zložitá.

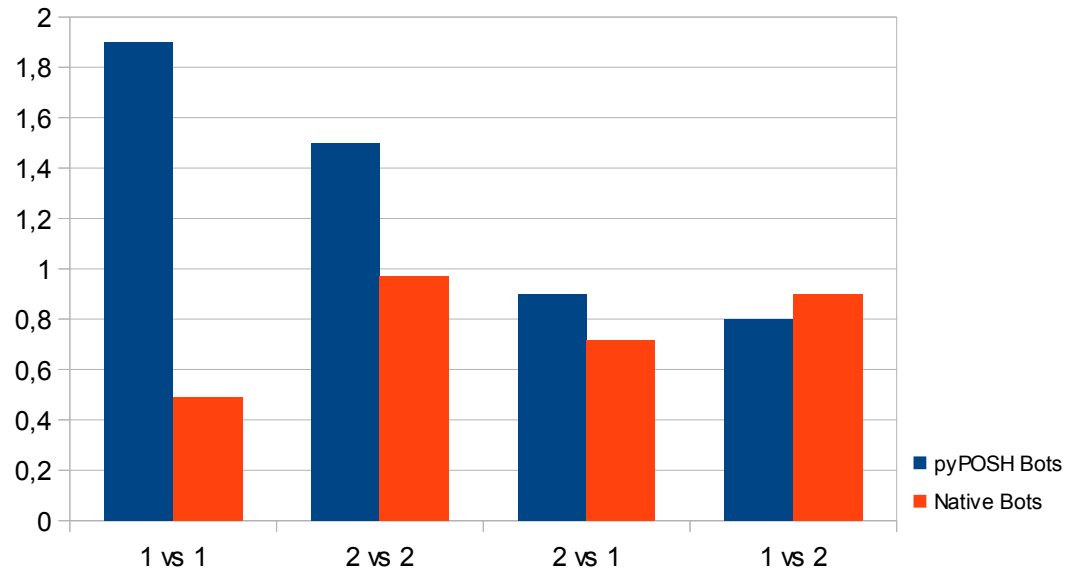
4.1 – Testovacie kritéria

V DM mode bola testovaná relatívna úspešnosť botov v jednotlivých kategóriach a to z dvoch rôznych pohľadov:

A) Testovanie porovnávaním - počet zabití k počtu smrti (K/D ratio)
B) Detekcia nechcených správaní botov na základe empirického pozorovania

4.2 – Výsledky A

Výsledky testovacej štatistiky K/D ratio obrázok 4. Pri dvojiciach sa vypočítal aritmeticky priemer ich K/D ratia.



Obrázok 11: K/D Ratio

Z výsledkov je možne usudzovať, že jeden na jedného pyPOSH bot obstal veľmi dobre a u 2 vs 2 je vidieť skoro 30% zlepšenie oproti klasickým agentom. Zbytok je pomerne vyrovnaný.

4.3 – Výsledky B

Pri sledovaní jednotlivých zápasov, som prišiel na chyby v pyPOSH pri exekúcii *action patternov*. I keď v súčasnom pláne niesu, tak vo fázy "pred-testovania" boli a ukázalo sa, že ich zahniezdením pod kompetencie vzniká vo vykonavaní pyPOSH plánu lokálny cyklus (nikdy nevyskočí spod kompetencie). Za (NE)úspechom pyPOSH botov nestojí samotný dizajn plánu, ale hlavne sofistikovanejšie algoritmy v moduloch shooting, memory. Demonštrovať to hlavne časti, kde bot volí lepšiu zbraň proti protivníkovi, ci kde si volí vhodnejšieho nepriateľa (vzdialenosť, život, zbraň).

5 Záver

Projekt pyPOSH aka TeamBOTs ukázal možnosť ako navrhovať umelú inteligenciu pre FPS počítačové hry. Definoval nové úrovne návrhu a to riešenie jednotlivých algoritmov (zavádzanie heuristík) akcii / zmyslov , a abstraktnejšiu v podobe dizajnu pyPOSH plánov. Zároveň odhalil nedostatky v súčastnej implementácii pyPOSH exekútoru. Príkladom môže byť redundancia návratovej hodnoty akcie *ActionResult.FINISHED*, tak ze vlastne neexistuje rozdiel s *ActionResult.RUNNING_ONCE*. V dokumentácii je definovaný *RUNNING_ONCE* ako príznak yaPOSHI na ukončenie akcie v nasledujúcom cykle vykonávania plánu a *FINISHED* ako okamžite ukončenie akcie ešte v danom cykle. Pogamut sa ukázal ako veľmi problemový vzhľadom na mavenizáciu – jednotlive knižnice importované priamo zo vzdialeného repozitára. Takéto riešenie značne obmedzuje samostatné fungovanie kódu. Je veľmi pracné vyhľadať všetky zdroje k použitým skompilovaným knižniciam a prerobiť importy tak aby to fungovalo v samostatnom projekte. S tým súvisí aj riešenie cez akési archetypy botov. Prácu s vývojom komplikuje aj nedostatočná JavaDoc dokumentácia pogamutu, ako malé chyby už v existujúcej + roztrúsenosť tutorialov. Príkladom chybnej dokumentácii je kolekcia TabooSet kde sa tato trieda sprava inak ako je v JavaDocu.

Avšak aj cez tuto pomalú krivku učenia pogamutu sa po jeho zvládnutí ukazuje ako skvelý nastroj na vedecku činnosť v oblasti tvorby zložitých virtuálnych agentov pre zložite realistické 3D prostredie.