VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

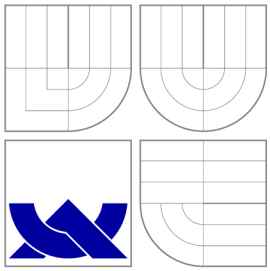
AGENT – GAMEBOT PRO UNREAL TOURNAMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

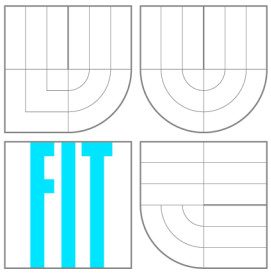
AUTOR PRÁCE
AUTHOR

MARTIN PŘÍBORSKÝ

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

AGENT – GAMEBOT PRO UNREAL TOURNAMENT

GAMEBOT FOR UNREAL TOURNAMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN PŘÍBORSKÝ
AUTHOR

VEDOUCÍ PRÁCE — Ing. RÓBERT KALMÁR
SUPERVISOR

BRNO 2014

Abstrakt

Úkolem této bakalářské práce je návrh a tvorba agenta-gamebota s modulární architekturou pro hru Unreal Tournament 2004. Hlavním cílem agenta bylo dosáhnout co nejlepších výsledků a v závěru obstát i proti nativním agentům hry. Výsledný agent je naimplementován v jazyce Java na platformě Pogamut.

Abstract

The goal of this bachelor's thesis is to design and implement a gamebot for Unreal Tournament 2004 game at Pogamut platform. The main aim was to achieve the best possible results and in the end stand against the native gamebots of the game. Final gamebot is based on Pogamut platform and written in Java language.

Klíčová slova

Agent, Unreal Tournament 2004, Pogamut, Umělá inteligence, Java.

Keywords

Gamebot, Unreal Tournament 2004, Pogamut, Artificial intelligence, Java.

Citace

Martin Příborský: Agent – Gamebot pro Unreal Tournament, bakalářská práce, Brno, FIT VUT v Brně, 2014

Agent – Gamebot pro Unreal Tournament

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Róberta Kalmára. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Příborský
31. července 2014

Poděkování

Tímto bych chtěl poděkovat vedoucímu práce Ing. Róbertu Kalmárovy za jeho čas, ochotu, cenné rady a připomínky.

Obsah

Kapitola 1

Úvod

Umělá inteligence (UI) je obor informatiky zabývající se tvorbou strojů nebo programů vykazující známky inteligentního chování.

Krom jiného hraje UI důležitou roli i v digitálních hrách. Nejčastěji tvorbou virtuálních agentů, specifičtěji nazývaných jako herní boti. Jedná se o slabý expertní systém, který ovládá umělého hráče nejčastěji ve FPS (First-person shooters) hrách. Účelem těchto počítačem ovládaných botů je simulování inteligentního lidského chování. Ve výsledku mohou tito boti hrát proti dalším botům, nebo skutečným hráčům.

Cílem této bakalářské práce je navržení a vývoj takového agenta na platformě Pogamut pro hru Unreal Tournament 2004 v jazyce Java.

1.1 Motivace

Motivací bylo zúčastnění se soutěže Pogamut Cup (`www.pogamutcup.com`). Jedná se o turnaj těchto počítačem ovládaných agentů (botů) v Unreal Tournament 2004 v módu deathmatch. Turnaj pořádá informatická sekce Matematicko-fyzikální fakulty Univerzity Karlovy v Praze přesněji výzkumná skupina AMIS (`artemis.ms.mff.cuni.cz`). Jedná se již o třetí ročník akce, která proběhne ve dnech 28. – 29. 6. 2014 [3].

1.2 Struktura práce

Samotný text práce je rozdělen na dva hlavní logické celky.

První část popisuje teorii a použité prostředky. V kapitole 2 jsou popsány základní teorie agentů a agentního prostředí. Kapitola 3 se věnuje hře Unreal Tournament 2004 a jsou v ní popsány všechny potřebné informace o ní. Modul GameBots2004 je popsán v kapitole 4. Informace o platformě Pogamut jsou následně popsány v kapitole 5.

Druhá část už je zaměřená na vývoj agenta. V kapitole 6 je popsán jeho návrh, v kapitole 7 jeho implementace a v kapitole 8 jsou popsány způsoby jeho testování. V závěrečné kapitole 9 jsou zhodnoceny dosažené výsledky a diskutovány možné budoucí rozšíření aplikace.

Kapitola 2

Teorie agentů

V této kapitole jsou popsány a vysvětleny typy agentů a agentního prostředí. Kapitola z větší části čerpá z [15] a [10]. Podkapitola 2.9 z [11].

2.1 Agent

Obecně je agent prvek systému vytvořený člověkem k nějakému předem zamýšlenému účelu. Cílem agenta je změna agentního systému do požadovaného cílového stavu. Obecně lze rozdělit agenty na:

- **Biologičtí agenti** – lidé
- **Techničtí agenti** – roboti
- **Programoví agenti** – softboti (agenti v počítačových hrách, počítačové viry)

Nejčastěji uváděnou společnou vlastností agentů je autonomie.

Agent je autonomní v tom smyslu, že je schopen, pokud je to vůbec možné, dosáhnout svých záměrů bez vnějších zásahů, tj. pouze interakcí s prostředím.

2.2 Inteligentní agent

Řeší úlohy inteligentním způsobem. Tím se myslí agentova schopnost plnit cíle, které jsou v jeho zájmu a to pomocí jeho vlastní "inteligence", ve většině případů logickou dedukcí. Agentovy záměry budou převážně souviset s účelem jeho vzniku. Agent je však nemusí mít vždy pevně zabudované, ale může je také přejímat od jiných agentů a pak se jimi řídit.

2.3 Reaktivní agent

Tento agent bezprostředně reaguje na jisté změny prostředí (nebo své změny vůči prostředí), aniž by měl vnitřní reprezentaci znalostí o tomto prostředí.

Jeho reakce nejsou výsledkem výpočtů či dedukcí na základě znalostí, ale pouze reakcemi na podněty. Neobsahuje žádné moduly pro tvorbu plánů ani moduly pro rozhodování, který cíl z potencionální množiny cílů bude sledovat.

Agent bývá vnitřně uzpůsoben z tzv. kompetenčních modulů, což jsou moduly, které se aktivují na základě přijatého stimulu a vykonávají určitou akci.

Reaktivita je vedle autonomity druhou významnou vlastností agentů.

2.4 Deliberativní agent

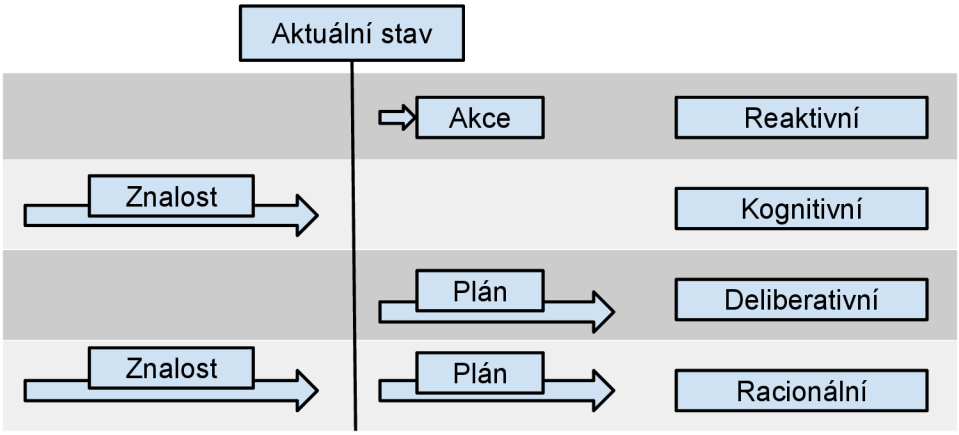
Na rozdíl od reaktivního agenta má deliberativní (rozvážný) agent schopnost plánovat postup svých akcí vedoucích k dosažení zvolených nebo zadaných záměrů/cílů. To znamená, že agent musí mít schopnost různých výpočtů (vnitřní činnost agenta). K dosažení svých záměrů pak agent ovlivňuje okolní prostředí tak, aby získal nějakou výhodu. Nedokáže však svobodně volit prostředky pro plnění cílů, postup jakým k cíli dojde, ani nemůže zlepšovat své chování či funkcionalitu.

2.5 Kognitivní agent

Kognitivní agent má schopnost vyvozovat logické závěry ze svých pozorování okolního prostředí. Takový agent musí především být schopen se učit a vytvářet si svou vlastní bázi znalostí. Do ní si během svého působení ukládá informace získané interakcí s okolím nebo znalosti získané dedukcí. Kognitivní agent nemusí mít nutně deliberativní schopnosti. Pak provádí pouze vnitřní akce, například analyzuje scénu, provádí překlad, nebo získává znalosti (dolování znalostí z dat).

2.6 Racionální agent

Racionální agent má všechny výše uvedená vlastnosti a jeho struktura obsahuje jak plánovací jednotku, tak i kognitivní jednotku včetně báze znalostí. Je to agent, který je na základě svých poznatků schopen se učit a pak plánovat svoji činnost tak, aby dosáhl svých cílů racionálním způsobem. Stojí nejvýše na pomyslné hierarchii uvedených agentů.



Obrázek 2.1: Rozdělení agentů Zdroj: [15]

2.7 BDI agent

Agent založený na modelu Belief-Desire-Intention, je zvláštní druh deliberativního či racionálního agenta, který se vyznačuje tím, že jeho jednání vychází z jeho představ (Beliefs), přání (Desires) a záměrů (Intentions).

- **Představy** – Informace, které agent má. Představy nemusí být pravdivé, ani statické.

- **Přání** – Čeho by chtěl agent dosáhnout. Přání mohou být krátkodobé i dlouhodobé (cíle). Agent nemusí být schopen všech dosáhnut a některé se mohou i vzájemně vylučovat.
- **Záměry** – Co se agent může rozhodnout dělat ke splnění přání a cílů.

2.8 Agentní prostředí

Prostředí je vše, s čím agent přichází během své činnosti do styku. Je to tedy agentní systém bez jediného svého prvku – agenta. Prostředí z hlediska agenta pak může být:

- **Plně pozorovatelné/Částečně pozorovatelné**

  Prostředí je pro agenta plně pozorovatelné, pokud může svými senzory sledovat jeho kompletní stav.

- **Epizodické/Sekvenční**

  Prostředí je epizodické, pokud budoucí vjemy nezávisí na předchozích vjemech a akcích.

- **Statické/Dynamické**

  Prostředí je pro agenta statické, pokud se může měnit pouze jeho akcemi.

- **Deterministické/Stochastické**

  Prostředí je deterministické, jestliže je jeho stav po vykonání nějaké akce dán pouze touto akcí a předcházejícím (původním) stavem tohoto prostředí.

- **Diskrétní/Spojité**

  Prostředí je diskrétní pokud má konečně nebo spočetně mnoho stavů.

- **Známé/Neznámé**

  Prostředí je známé, pokud agent zná všechny jeho funkce a možnosti využití.

2.9 Boti v FPS hrách

Boti jsou virtuální lidé (ztělesnění virtuální agenti) ve FPS hrách. Tito boti se pohybují, střílí, bojují a musí být řízeni. Protože existuje mnoho přístupů v oblasti UI, existuje mnoho způsobů, jak dokončit úkoly řízení bota. Tyto přístupy se liší v jejich použitelnosti pro řešení různých typů problémů, jejich univerzálnosti a také v jejich výpočetní složitosti. Většina společných přístupů (s ohledem převážně na UI botů) jsou:

- **Skriptovaná UI** (používá se ve starších počítačových hrách, lze se dohadovat, zda se skutečně jedná o UI)
- **Konečné automaty** (v současných hrách se používají takřka výhradně, díky své univerzálnosti)
- **If-then pravidla** (jednoduchý přístup, blízké konečným automatům, ty jsou často realizovány s if-then pravidly)
- **Fuzzy plánování** (používá se spíše v akademické sféře)

- **Neuronové sítě** (můžou být použity pro učení bota, nebo některé dobře definované problémy)

Zajímavým termínem je i "podvádění UI". Tím je v digitálních hrách myšleno poskytovat agentovy informace, které by byly nedostupné skutečnému hráči ve stejné situace [13], například informace o přesné pozici ostatních hráčů.

Často je však toto podvádění používáno úmyslně, pokud je totiž UI agenta něčím omezena, mnohdy je to jediný způsob, jak zajistit, aby se takovýto agent vyrovnal lidským protivníkům.

Kapitola 3

Unreal Tournament 2004

Hra Unreal Tournament 2004, jinak známa i pod zkratkou UT2004 nebo UT2K4, je multiplayerová FPS s částečně otevřeným kódem, vyvinuta společnostmi Epic Games a Digital Extremes.

Hra UT2004 je podle rozdělení v podkapitole 2.8 prostředí částečně pozorovatelné (hráč/bot vidí jen pohledem z první osoby, nemá informace o zbytku mapy), sekvenční, dynamické, slabě stochastické (například při střílení není 100 % pravděpodobnost zásahu), spojité a slabě neznámé (programátor nemusí znát všechny možnosti, které lze použít ve hře). Hlavně z důvodu poslední vlastnosti, je dobré se s hrou, jejím prostředím a všemi jejími možnosti dobře seznámit, čemuž je věnována tato kapitola.

Slouží tedy především k seznámení se s módy hry a způsoby pohybu. Obsahuje podrobněji popsané druhy a vlastnosti zbraní, předměty, které lze ve hře sbírat a adrenalinová komba, které lze ve hře využít.

Tato kapitola vychází z uživatelského manuálu ke hře [4], stránek [1], [2], [14] a vývojářské Pogamut wiki [12].

3.1 Módy hry

Ve hře existuje 10 různých módů souboje:

- **Deathmatch** – Všichni hráči jsou postaveni proti všem ostatním. Vyhrává ten, kdo má za hru největší počet bodů (počet bodů = počet zabití ostatních hráčů - počet zabití sebe samotného). Spadají pod něj i další dvě modifikace:
  - **InstaGib** – Zde má každý hráč k dispozici pouze jednu speciální zbraň, která má nekonečno nábojů a zabíjí jediný zásah (bez ohledu na to kolik má nepřítel zdraví).
  - **Team Arena Master** – Hráč na začátku kola dostane většinou všechny zbraně a polovinu nábojů k nim. Na mapě pak už žádné další zbraně, náboje ani lékárničky nejsou a musí si tedy vystačit s tím, co má.
- **Team death match** – Hráči jsou rozděleni do týmů (typicky modrý a červený tým), nezáleží na výkonu jednotlivce, ale na celém jeho týmu. Vyhrává tým s největším počtem bodů.

- **Capture the flag** – Cílem obou týmů je získat vlajku z území druhého týmu a vrátit se s ní k vlajce na svém území. Pokud je nositel vlajky zabit, vlajka zůstává na místě, kde ji může vzít jiný hráč opačné barvy a pokračovat v přenášení. Pokud ji vezme tým stejné barvy, vlajka se vrací zpět na své území.
- **Onslaught** – Úkolem obou týmu je zabírat strategické body na mapě až k základně nepřítele. Útok na základnu je znemožněn, pokud nejsou zabrány strategické body v jeho blízkosti. Vítězí ten, kdo jako první zničí jádro nepřítele umístěné na jeho základně. Hraje se na rozsáhlých strategických mapách, kde jsou k dispozici také vozidla, obrané věže a tři dodatečné zbraně.
- **Assault** – Hrají dva týmy rozdělený na útočníky a obránce. Útočníci musí splnit požadované úkoly do určitého časového limitu. Obránci se jim v tom snaží zabránit. Po každém kole se role vymění. Pokud se útok povede oběma týmům, vyhrává ten, který ho provedl rychleji.
- **Double domination** – V tomto módu je úkolem obou týmů na deset sekund obléhat dva různé body na mapě (bod patří tomu týmu, jehož hráč přes něj přeběhl naposledy). Po úspěšném obležení se body stanou neutrálními na dalších deset sekund. Většinou se hraje na dvě až tři vítězná obležení.
- **Bombing run** – Cílem je ukořistit balón uprostřed mapy a vhodit ho do nepřátelské brány. Tým získává 3 body, pokud balón bránou prohodí a 7 bodů, pokud ho drží a sám bránou proskočí. Hráč nesoucí balón nemůže používat zbraně, ale může ho přehodit jinému hráči, pokud je tento hráč zabit, balón zůstává na zemi, kde ho může sebrat jakýkoli jiný hráč.
- **Last man standing** – Všichni hráči jsou postaveni proti sobě, každý má všechny typy zbraní s maximálním počtem nábojů, 100 bodů štítu, ale omezený počet životů. Poslední hráč na živu vyhrává.
- **Invasion** – Všichni hráči hrají společně proti různorodým monstrům, přicházejících v jednotlivých vlnách. Každá vlna trvá 90-240 sekund. Pokud hráč během vlny zemře, zůstává mrtvý do konce vlny. Vlnu musí přežít aspoň jeden z týmu, aby byla splněna, jinak hra končí. Úkolem je vydržet co nejvíce vln, jejichž složitost se postupně zvětšuje.
- **Mutant** – Na začátku se jeden hráč stane mutantem, tento hráč získává všechny zbraně, velké množství nábojů, neviditelnost, větší rychlost pohybu a střelby. Za to mu však pomalu ubývají životy, které se obnovují pouze zabíjením jiných hráčů. Navíc všichni ostatní vidí přibližnou pozici mutanta na mapě a snaží se ho zabít. Ten, komu se to podaří, se jím následně stává. Vyhrává ten, komu se vydrží zůstat jako mutant nejdéle.

3.2 Pohyblivost

Kromě obyčejného pohybu po mapě existují ve hře i další způsoby pohybu, jejímž použitím se dá dosáhnout na jinak nedosažitelná místa, nebo se dají efektivně využít při souboji.

- **Normal Jump** – Obyčejný skok.
- **Double Jump** – Dvojitý skok.

- **Lift Jump** – Skok za použití výtahu. Při správném časování se rychlost výtahu jedoucího nahoru zkombinuje se skokem a vymrští hráče do jinak nedosažitelné výšky.
- **Dodge** – Je to speciální pohyb (úskok), který lze udělat rychlým zmáčknutím klávesy dvakrát po sobě, tím se docílí nízkého, rychlého skoku, který je ideální pro vyhýbání vystřelených projektilů, nebo krytí (úskokem za překážku).
- **Dodge Jump** – Úskok kombinovaný se skokem, je vyšší a delší. Je vhodný používat k rychlému pohybu po mapě (rychlejší než obyčejná chůze), ale delší doba ve vzduchu dělá z hráče snadnější cíl.
- **Wall Dodge** – Je-li hráč poblíž zdi, je možné se od ní odrazit pomocí dodge. To se dá efektivně využít například při pádu. Opět lze kombinovat se skokem (Wall Dodge Jump).
- **Shield Jump** – Je to speciální skok pomocí zbraně shield gun. Hráč tak může skákat až neuvěřitelně daleko/vysoko, ovšem za cenu snížení počtu životů sebe samotného. Obdobně lze skákat i za použití jiných zbraní s explozními projektily (sekundárním módu zbraně shock rifle, rocket luncher, aj.).

3.3 Zbraně

Každá zbraň má dvě funkce, primární a sekundární střelbu.

- **Shield Gun** – Základní zbraň, nelze odhodit, nemá klasické náboje, při používání se postupně vybíjí, nabití pak nějakou dobu trvá.
  - Primární útok: dá se zaútočit kliknutím (40 poškození za 0.55 s), nebo držením tlačítka nabít a útok je tak silnější (150 poškození pokud je plně nabito za 2.27 s). Plně nabitý Shield Gun ultimativně odstraní i protivníka s plným štítem, pokud se k němu dostanete, což ovšem není jednoduché, nejlépe se toho dá docílit Dodge jumpem. Útok se vypouští automaticky při dotyku soupeře.
  - Sekundární útok: vytvoří obranný štít, který se postupně vybíjí. Štít chrání před většinou útoků, některé projektily odráží zpátky. Štít se při absorbování ran vybíjí rychleji. Lze použít i pro absorbování poškození při pádu z velké výšky.
- **Assault Rifle** – Základní zbraň, lze odhodit. Má slabou útočnou sílu, ale pokud hráč sebere druhou, může používat obě, což útočnou sílu zdvojnásobí. Na velkou dálku je nepřesná. Druhým módem zbraně je granátomet. Čím déle se podrží tlačítko, tím dál se granát vystřelí, vybuchuje při dotyku soupeře.
  - Primární útok: 7 poškození na kulku, okamžitý zásah, 6.88 kulek za s.
  - Sekundární útok: 70 poškození na granát, 0.91 s na výstřel, dalších 0.91 na opětovné nabití.
- **Shock Rifle** – Primární paprsek má velmi dobrý dostřel, ale malou kadenci. Sekundární útok vystřelí pomalejší světelnou kouli, za to má kadenci lepší. Když trefíte primárním útokem kouli (i soupeřovu), vznikne exploze.
  - Primární útok: 45 poškození na zásah za 0.64 s.
  - Sekundární útok: 5 až 45 poškození, podle blízkosti za 0.55 s.

  - Combo útok: Teoreticky až 200 poškození, většinou pod 150, podle blízkosti výbuchu.

- **Link Gun** – Primární útok střílí plazmové projektily. Sekundární plazmový laser, který zabíjí na krátkou vzdálenost. Lze jím i opravovat vlastní vozidla, nebo znásobovat poškození střelby spoluhráče, pokud tuto zbraň také používá.
  - Primární útok: 30 poškození na projektil, vystřelí 5.5/s.
  - Sekundární útok: 9 poškození na 0.12 sekund přímého působení, může brzdit pohyb nepřítele.

- **Bio-Rifle** – Střílí radioaktivní hmotu, pokud nezasáhne soupeře, zůstává pár vteřin na místě dopadu a exploduje, pokud se k ní protivník přiblíží. Vhodné pro obranu a zablokování cesty pří útěku před soupeřem. Sekundárním útokem se hromadit větší dávka, projektil se stane větším silnějším, ale pomalejším.
  - Primární útok: 35 poškození na přímý zásah za 0.3 s.
  - Sekundární útok: 35 až 210 při maximálním nabití za 2.05 s.

- **Minigun** – Ruční rotační kulomet. Primární útok je rychlejší, slabší a méně přesný na dálku, sekundární je pomalejší, ale silnější a přesnější i na větší vzdálenost.
  - Primární útok: 7-8 poškození na kulku, 16.5/s.
  - Sekundární útok: 14-16 poškození na kulku, 5.5/s.

- **Flak Cannon** – Primární střelba odpálí granát přímo v hlavni zbraně, takže všechny šrapnely (střepiny granátu) letí přímo vpřed s velkým rozptylem (jako brokovnice). Šrapnely se odráží od stěn, takže můžete zranit nepřítele, i když je za rohem. Sekundárně vystřelí dělostřelecký granát, jenž letí po balistické křivce a při dopadu se rozletí ve velké množství šrapnelů. Díky balistickému klesání projektilu lze střílet přes římsy, aniž by hráče nepřítel stojící na nich nebo pod nimi viděl.
  - Primární útok: 13 na projektil, jeden výstřel - 9 projektilů * 13 = 117 poškození na plný zásah. Poškození je snižováno o 1 každých 0.2 sekund, minimální poškození 5 na projektil. Rychlost střelby 1.23/s.
  - Sekundární útok: 90-168 poškození (vysoce náhodné) při přímém zásahu, jinak poškození výbuchem, dle vzdálenosti. Rychlost střelby 0.99/s.

- **Rocket Launcher** – Raketomet, střílí jednotlivé rakety, sekundární útok nabije a vystřelí až 3 rakety naráz, které letí vedle sebe a zasáhnout tak více míst. Pokud se před vypuštěním nabitých raket přidrží primární střelba, letí v těsné spirálové formaci a výbuch všech je soustředěn na jedno místo. Když hráč udrží nepřítele na několik sekund v zaměřovači, jsou vystřelené rakety naváděné (a doženou ho i za rohem). Dá se jim ale uhnout úskokem do strany.
  - Primární útok: 90 poškození při přímém zásahu, jinak poškození výbuchem, dle vzdálenosti, rychlost střelby 1.22/s.
  - Sekundární útok: primární útok násobený číslem 2 nebo 3, podle počtu vypálených raket, rychlost nabíjení 1.16/s.

- **Sniper Rifle** – Klasická odstřelovací puška.

  - Primární útok: 60 poškození na zásah, 125 poškození na zásah do hlavy nepřítele, za 1.21 s.
  - Sekundární útok: přiblížení.

- **Lightning Gun** – Blesková puška, s funkcí přiblížení.
  - Primární útok: 70 poškození na zásah, 140 na zásah do hlavy nepřítele, za 1.5 s.
  - Sekundární útok: přiblížení.

3.4 Předměty

V UT2004 jsou různé předměty, které může hráč sbírat. Nejčastěji jde o předměty zvyšující ukazatel zdraví, nebo štítu.

Hráč začíná se zdravím na hodnotě 100 a štítem na hodnotě 0. Zdraví lze navýšit až do hodnoty 199, štít do hodnoty 150.

- **Health Pack** – Zvýší zdraví o 25, maximálně do celkové hodnoty 100.

  Obnovení předmětu trvá 27.27 sekund.

- **Keg o' Health** – Zvýší zdraví o 100, maximálně do celkové hodnoty 199.

  Obnovení předmětu trvá 54.54 sekund.

- **Health Vial** – Zvýší zdraví o 5, maximálně do celkové hodnoty 199.

  Obnovení předmětu trvá 27.27 sekund.

- **Shield Pack** – Zvýší štít o 50, maximálně do celkové hodnoty 50.

  Obnovení předmětu trvá 27.27 sekund.

- **Super Shield Pack** – Zvýší štít o 100, maximálně do celkové hodnoty 150.

  Obnovení předmětu trvá 54.54 sekund.

- **Double Damage** – Dvojnásobné poškození (při střelbě) po dobu 27.27 sekund.

  Obnovení předmětu trvá 81.81 sekund.

- **Adrenaline Capsule** – Přidá 3 adrenalinové body, více v následující podkapitole 3.5.

  Obnovení předmětu trvá 27.27 sekund.

3.5 Adrenalinová komba

Sebrání ampulky přidá tři adrenalinové body, zabití nepřítele pět. Jakmile hráč dosáhne stovky adrenalinových bodů, je možné použít některé adrenalinové kombo, které dává hráči dočasnou výhodu nad soupeřem. Vyvolá se stiskem 4 specifických kláves.

- **Speed** – Zvýší rychlost hráče na dvojnásobek.
- **Booster** – Zvyšuje postupně zdraví/štít až na 199/150.
- **Invisible** – Neviditelnost.

- **Berserk** – Dvojnásobná rychlost střelby.
- **Super jump** – Dvojnásobná výška skoku.

Kapitola 4

Projekt GameBots

Tato kapitola popisuje modul GameBots, použitý ve hře UT2004 k ovládání agentů. Kapitola čerpá z [7] a [8].

4.1 Původ

Projekt GameBots vytvořil Andrew N. Marshal a Gal Kaminka na půdě ústavu informačních věd Univerzity Jižní Kalifornie. GameBots zprvu používal starší verzi hry – Unreal Tournament (2000). Jejich cílem bylo umožnit používání prostředí hry pro výzkum v oblasti umělé inteligentce.

V projektu pokračovali Joe Manojlovich, Tim Garwood a Jessica Bayliss z univerzity RIT (Rochester Institute of Technology). Ti přenesli projekt na novější verzi hry – UT2004.

GameBots je nyní modifikace pro UT2004, sloužící primárně pro vytvoření bohatého prostředí k vývoji virtuálních agentů. Tato modifikace poskytuje síťový TCP/IP protokol k získávání informací o prostředí a pro kontrolu herních postav. Je naprogramována v skriptovacím jazyku UnrealScript – staticky/silně typovaný objektově orientovaný, událostmi řízený programovací jazyk, který byl vytvořen a používán pro UT2004 herní mechanismy (skoro pro všechny, kromě grafického enginu). UnrealScript je podobný Javě, nebo jazyku C++ [6].

4.2 Textový protokol

Textový protokol obsahuje příkazy a zprávy, obojí sdílí stejný textový formát. Příklad GameBot zprávy:

```
PLAYER {Name GoodBot} {Location 1400,500,0} {Rotation 32000,0,0} {Health 100}
{Height 170.59} {IsHuman False}
```

První řetězec značí typ zprávy. Poté uzavřený párem složených závorek následují dvojice atributu. Jedná se o název atributu a hodnotu atributu oddělené mezerou.

Správný rozbor výše uvedené zprávy by tedy byl:

- Typ zprávy – `PLAYER`
- Atribut 1 – Název atributu je "`Name`". Hodnota atributu je "`GoodBot`".

- Atribut 2 – Název atributu je "`Location`". Hodnota atributu je "`1400,500,0`".
- Atribut 3 – Název atributu je "`Rotation`". Hodnota atributu je "`32000,0,0`".
- Atribut 4 – Název atributu je "`Health`". Hodnota atributu je "`100`".
- Atribut 5 – Název atributu je "`Height`". Hodnota atributu je "`170.69`".
- Atribut 6 – Název atributu je "`isHuman`". Hodnota atributu je "`False`".

Počet atributů zprávy není omezen. Pro příkazy už limit existuje – pro Pogamut GameBots je to 32 atributů.

4.3 Datové typy

Typy dat používané GameBots jsou běžné datové struktury UnrealScriptu.

Nejpoužívanější typy jsou přirozená, nebo realná čísla (atributy `Health` a `Height`), řetězce (atribut `GoodBot`), boolean (`atribut isHuman`) a vektorové a úhlové struktury (atributy `Location` a `Rotation`).

Pro měření polohy a úhlů v UT2004 byly definovány tzv. unreal jednotky. Jedna unreal jednotka odpovídá zhruba 1 cm. Pro úhly, plné otočení (o 360 stupňů) odpovídá 65535 unreal jednotek. Pro převod na radiány lze jednoduše vydělit číslem 65535 a vynásobit $2 * \pi$.

Vektorové struktury jsou obvykle používány pro definování pozice (v našem případě atribut `Location`). Vektor je trojnásobný, definující `x`, `y` a `z` souřadnici v UT2004 prostředí.

Rotační struktury se používají k měření otáčení objektů (v našem případě atribut `Ratation`). Rotační struktury jsou také trojnásobné, skládající se ze stoupání, stáčení a válení. Stoupání udává rotaci nahoru/dolů, stáčení rotaci vlevo/vpravo a válení je ekvivalent dělání hvězdice.

4.4 Typy připojení

V GameBots jsou tři typy připojení:

- **Připojení bota** – použitý pro vytvoření a kontrolování bota v prostředí a ve stejném čase pro získávání informací o okolí bota (věci které vidí a vnitřní stavy - zdraví, brnění, tým atd.).
- **Kontrolní připojení** – použitý pro kontrolování herních mechanizmů. Může pozastavit hru, změnit mapu, vyhodit hráče ze hry a více.
- **Pozorovací připojení** – používá se pro pozorování bota nebo hráčů ve hře. Informace o tom, co hráč nebo bot dělá, se exportují přes standartní GameBots zprávu.

Pro inicializace komunikace s GameBots musí být splněny tyto tři kroky:

- Spustit UT2004 server s GameBots.
- Připojit se k UT2004 přes TCP/IP na port 3000 (defaultní port pro připojení bota), 3001 (defaultní port pro kontrolní připojení), nebo 3002 (defaultní port pro pozorovací připojení).

- Inicializovat komunikaci s GameBots pomocí handshakingu (mírně rozdílný pro druhy připojení).

Kapitola 5

Platforma Pogamut

Pogamut je middleware psaný v Javě, který umožňuje kontrolovat virtuální agenty ve více prostředích poskytovaných herními enginy.

V současné době podporuje hry UT2004, Unreal Engine 2 Runtime Demo, Unreal Development Kit a DEFCON.

Jelikož bude výsledný bot implementován na této platformě, tato kapitola slouží k bližšímu seznámení se s ní, informace pochází z [9].

5.1 Vlastnosti

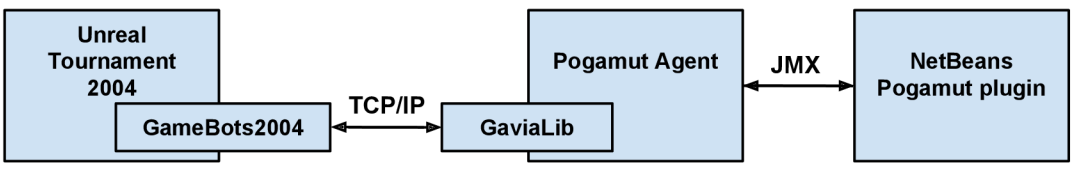
Platforma je úzce svázána s virtuálním světem hry UT2004, začleněná do vývojářského prostředí s podporou pro ladění chyb. Dále umožňuje připojení do reaktivního plánovače POSH pro ovládání chování agentů obsahující i vizuální editor pro toto POSH plánování a mnoho dalšího.

Hlavní výhodou je, že platforma umožňuje rychlé psaní kódu virtuálního agenta a řeší řadu nízko úrovňových problémů, jako např. připojení k prostředí, exportování informací o virtuálním světě, nebo poskytování hledání základních tras pohybu.

Většinu akcí tak lze v prostředí provést jedním, nebo dvěma jednoduchými příkazy. Díky tomu se dá při psaní kódu soustředit spíše na vylepšování algoritmů agenta.

5.2 Architektura

Technicky se Pogamut skládá z pěti hlavních částí (viz. obr. 5.1). UT2004 do které je přidán modul GameBots2004. Ten komunikuje pomocí TCP/IP s rozhraním knihovny Gavialib, která přetváří zprávu do struktur samotnému Pogamut agentovy, který z nich čerpá informace o stavu prostředí ve hře. Poslední částí je pak Pogamut plugin pro vývojové prostředí NetBeans. Ten dokáže vzdáleně sledovat a ladit agenta pomocí protokolu JMX.



Obrázek 5.1: Obecné schéma architektury Pogamut

5.3 Knihovna GaviaLib

Knihovna byla vyvinuta pro všeobecné využití připojování agentů k téměř jakémukoli virtuálnímu prostředí. Jde o obecné rozhraní k široké škále takovýchto světů. Jsou po nich požadovány jen mírné nároky, prostředí musí umět pracovat s objekty a být schopno poskytovat informace pomocí událostí.

GaviaLib poskytuje:

- Abstraktní implementaci agenta, která nese jeho životní cyklus, umožňuje dálkové ovládání agenta přes standardní Java JMX protokol pro vzdálený přístup k programům a definuje společný soubor záznamů.
- Rozhraní agenta v prostředí, které spravuje objekty a události a informuje agenta o důležitých objektech událostí (např. zmizel/objevil se objekt).
- XML/XSLT Framework pro vymezení všech objektů a událostí prostředí.

5.4 Vývojové prostředí

Platforma Pogamut obsahuje zásuvný modul pro vývojové prostředí NetBeans, který může komunikovat s běžícími agenty přes JMX.

Prostředí nabízí několik užitečných vlastností pro návrháře, jako prázdné šablony projektu, ukázky agentů, spravování UT2004 serverů, seznam běžících agentů, rychlý přístup k obecnému nastavení agentů, ladění krokováním, zobrazení záznamů a manuální ovládač pozice agenta.

Kapitola 6

Návrh agenta

Agent má omezené informace o prostředí (podkapitola 6.1), pohybuje v dynamickém prostředí a musí tedy reagovat na všechny podměty a změny prostředí okamžitě. Tyto změny prostředí zachycují posluchače událostí, na které reaguje. Takové události a reakce na ně jsou popsány v podkapitole 6.2.

Zbylé rozhodování se provádí periodicky každých 250 ms. Spadá sem navigace po mapě popsána v podkapitole 6.3 a v případě kontaktu s nepřítelem i výběr soubojového módu 6.4. Při souboji taktéž hraje roli výběr zbraně popsán v podkapitole 6.5.

6.1 Informace

O sobě samotném má agent k dispozici aktuální a přesné informace. Jelikož se však pohybuje v částečně pozorovatelném prostředí, nezná tyto data o soupeři.

Z toho důvody je vytvořena strukturu uchovávající informace o soupeři, tj. odhadovaná hodnota jeho zdraví, brnění, zbraně, náboje a někdy i pozici. Tyto informace nejsou přesné a vycházejí ze znalostí, které agent má. Například ví, že soupeř začíná ze 100 životy a při útoku na něj dokáže zjistit poškození, které mu způsobil, a dle chybějících předmětů na mapě, dokáže zjistit, čím soupeř disponuje.

Přesné události, které k tomu využívá jsou uvedeny v následující podkapitole. Tyto informace jsou pak použity ve vyhodnocení souboje, tj. kdy je dobré na soupeře zaútočit a kdy je lepší se mu vyhnout. S tím souvisí podkapitola 6.4.

6.2 Události

Jak již bylo zmíněno, události slouží k registraci aktuálních změn a reaktivním akcím. Následuje výčet navržených událostí a jejich reakce na ně. Implementační detaily samotných posluchačů a druhů událostí jsou popsány v následující kapitole.

- Objevení soupeře (`playerAppeared`) – Soupeř vstoupil do zorného pole agenta. Při této události se nastaví zaměření na soupeře a zahájí střelba.
- Zmizení soupeře (`playerDisappeared`) – Soupeř zmizel ze zorného pole agenta, ukončit střelbu.
- Poškození soupeře (`playerDamaged`) – Použité pro výpočet odhadu soupeřova zdraví. Událost obsahuje hodnotu poškození, která se v struktuře obsahující informace o soupeři odečte od hodnoty předpokládaného zdraví soupeře.

- Poškození agenta (`botDamaged`) – Událost použita pro kontrolu, zda má agent soupeře v zorném poli, pokud ne, otočí se, aby ho nalezl.
- Zabití soupeře (`playerKilled`) – Použito pro reset informací ve struktuře o soupeři.
- Zabití agenta (`botKilled`) – Po smrti se uchovají informaci o posledním pobytu soupeře. Lze použít pro vyhnutí možného kontaktu s nepřítelem do doby, než agent nasbírá dostatek předmětů a bude tak lépe připraven pro další kontakt se soupeřem.
- Uslyšení zvuku (`hearNoise`) – Otočení směrem ke zdroji zvuku, z informace o vzdálenosti a směru odvození přibližné polohy soupeře.
- Zvednutí předmětu (`itemPickedUp`) – Agent zvedl předmět, uchová informace, že je předmět pryč a nastaví se čas znovuobjevení pro opětovnou možnost sebrat předmět.
- Zmizení předmětu (`itemPickedUp`) – V zorném poli agent zpozoroval chybějící předmět, který nesebral on. Provede se přičtení hodnoty předmětu k informacím vedeným o soupeři.
- Objevení projektilu (`incomingProjectileAppeared`) – Zpozorován blížící se projektil. Použito pro úskok před projektilem.

6.3 Navigace

Primárně se používá k získávání potřebných předmětů. Jak přesně navigace funguje je řešeno v následující kapitole. Důležité je vědět, že agent zná rozmístění předmětů po mapě a dokáže vyhledat nejbližší předmět potřebného typu. Zbývá mu tedy rozhodnutí ke kterému předmětu se vydat. To tvoří dle priorit předmětů a svým současným požadavkům.

Priorita navigace k dalšímu bodu je ukázána na pseudokódu 6.1:

```
if (agent.zivoty<75) navigaceLekarnicka();
if (agent.stit<99) navigaceStit();
if (doubleDamageExistuje()) navigaceDoubleDamage();
if (!maDobrouZbran()) navigaceZbran();
if (maloNaboju()) navigaceNaboje();
else navigaceNejblizsiPredemet();
```

Ukázka kódu 6.1: Pseudokód výběru navigace

Toto rozložení platí jen při vybírání hlavního cíle, pokud je některý předmět blízko určené cesty, vezme ho agent po cestě, i když pro něj není zrovna prioritní.

6.4 Soubojové módy

Slouží k rozhodování při kontaktu s nepřítelem. Využívá strukturu odhadovaných informací o soupeřovy popsané v podkapitole 6.1.

- Útok (`attack`) – V tomto módu agent utočí na soupeře, ale stále se pohybuje po své trase a pokud se mu soupeř ztratí z dohledu, nesnaží se ho pronásledovat.

- Pronásledování (`chase`) – Agent pronásleduje soupeře, i když se pokusí uniknout. K tomuto dochází, pokud má agent nad soupeřem výhodu. Konkrétně pokud má aspoň jednu zbraň, kterou považuje za kvalitní (pole `GOOD_WEAPONS` obsahující seznam nejúčinnějších zbraní), dostatek zdraví (80 a více), nebo pokud odhaduje, že má soupeř málo zdraví (40 a méně) a nekvalitní zbraň, či pokud zrovna sebral předmět "Double Damage".

- Ústup (`Retreat`) – Agent vyhledá nejbližší únikový bod a pokusí se uniknout. K tomuto scénáři dochází, pokud má soupeř nad agentem výhodu. Konkrétně pokud má málo zdraví (40 a méně), nebo žádnou zbraň, kterou by považoval za kvalitní, či pokud soupeř sebral předmět "Double Damage".

6.5 Výběr zbraně

Výběr zbraně se provádí dle její efektivity, vzdálenosti protivníka, okolního prostředí a zbraně protivníka.

1. Dle vzdálenosti je preference rozdělena do tří skupin:

   - Krátká vzdálenost (0-5m) – Zbraně efektivní na krátkou vzdálenost, bez zbraní způsobující plošné poškození (výbuch), který by kromě soupeře zraňoval i samotného hráče.
   - Střední vzdálenost (5-20m) – Zbraně efektivní na střední vzdálenost, i se zbraněmi způsobující plošné poškození.
   - Velká vzdálenost (20-100m) – Zbraně efektivní na velkou vzdálenost.

   Každá ze tří skupin má jinak prioritně seřazený seznam zbraní. V souvislosti se vzdáleností protivníka je z agentem vlastněných zbraní vybrána zbraň s nejvyšší prioritou.

2. Výběr podle soupeřovy zbraně.

   Projektily některých zbraní, lze jinou zbraní sestřelit, nebo i odrazit (viz. Podkapitola 3.3). Dle aktuální soupeřovy zbraně se změní agentova priorita výběru zbraně.

   V případě obrany je pak nutné rozlišovat, zdali protivník drží zbraň s projektily, kterým lze uskočit (například Rocket Launcher), nebo s efektem okamžitým, proti kterému je jediná možnost obrany zbraň "Shield Gun".

3. Výběr zbraně dle okolí.

   Okolní stěny hrají roly při použití zbraní s odrážejícími se projektily (primární útok zbraně "Flak Cannon"). Pokud by byla zbraň vystřelena naproti stěně, nebo jsou stěny moc blízko sebe, mohli by odražené projektily zasáhnout samotného agenta. Pokud se však soupeř snaží uniknout, lze ho výstřelem do stěny pod správným úhlem zasáhnout, i "za rohem", kde není aktuálně viditelný a kde by jinou zbraní nebyla možnost jeho zásahu.

   Dále hraje možnost i vyvýšení. Sekundárním útokem stejné zbraně lze třeba po balistické křivce střílet přes římsy, aniž by agenta soupeř stojící pod nimi viděl. Stejně tak lze však střílet po balistické křivce i směrem nahoru.

Kapitola 7

Implementace

Tato kapitola popisuje způsob implementaci výsledného agenta. Základní metody jsou popsány v podkapitole 7.1. Navigace v podkapitole 7.2 a události v podkapitole 7.3 Při implementaci bylo využito některých Pogamut ukázkových šablon.

7.1 Základní metody

Nejdůležitějších metod hlavní třídy `MyBot`, v pořadí dle volání metod při spuštění:

- `prepareBot(UT2004Bot bot)` – příprava agenta, volá se před spojením bota s GB2004 a spuštění v UT2004, obsahuje hlavně inicializace jednotlivých modulů agenta, detektor "zaseknutí"agenta a základní preferenci výběru zbraní.
- `getInitializeCommand()` – inicializace agenta, nastavení jména, vzhledu a přesnosti míření. Tato metoda je demonstrována na ukázce kódu 7.1:

```
public Initialize getInitializeCommand() {
return new Initialize().setName("MyBot").setDesiredSkill(6);
}
```

Ukázka kódu 7.1: Ukázka metody `getInitializeCommand()`

- `botInitialized(GameInfo info, ConfigChange config, InitedMessage init` – inicializace bota, volá se, když byl bot pozván do GB2004, ale ještě nevstoupil do hry. To znamená, že algoritmus podání ruky mezi agentem a serverem proběhl úspěšně a čeká se na další příkazy.
  Je zde k dispozici objekt `info`, který obsahuje informace o typu hry, mapě, apod. Dále objekt `config`, obsahující informace o současné konfiguraci – nastavení zpoždění viditelnost, apod. A nakonec objekt `init`, držící informace o proměnných jako je například jeho rychlost v prostředí, maximální možné zdraví apod.
- `beforeFirstLogic` – metoda, která se provede pouze před prvním zavoláním metody `logic()`, zde jsou prováděny poslední přípravy vlastních modulů. Bot je už v době volaní plně inicializovaný a přítomen v prostředí

- `logic` – hlavní metoda, která se volá periodicky většinou každých 250 ms. Metoda neustále ovládá agenta a tvoří rozhodnutí, co udělat příště, oředevšínm v oblasti navigace a souboje.

Některé další zajímavé metody třídy `MyBot`:

7.2 Navigace

Navigace byla provedena dle navigačního grafu (konkrétně pomocí třídy `UT2004Navigation`).

Navigační graf je základní reprezentace prostoru používaná pro navigaci pohybu agenta po mapě. Je tvořen uzly nazývanými navigační body reprezentující místo, kterého může agent dosáhnout. Pokud existuje přímá cesta z jednoho navigačního bodu do druhého, je spojena hranou.

Cesta z libovolného navigačního bodu A do navigačního bodu B je jednoduše řešena algoritmem pro vyhledávání optimální cesty v grafu (A* algoritmem). Výstupem algoritmu plánování cesty je posloupnost uzlů, kterými se musí projít, aby se dostalo do cílové destinace.

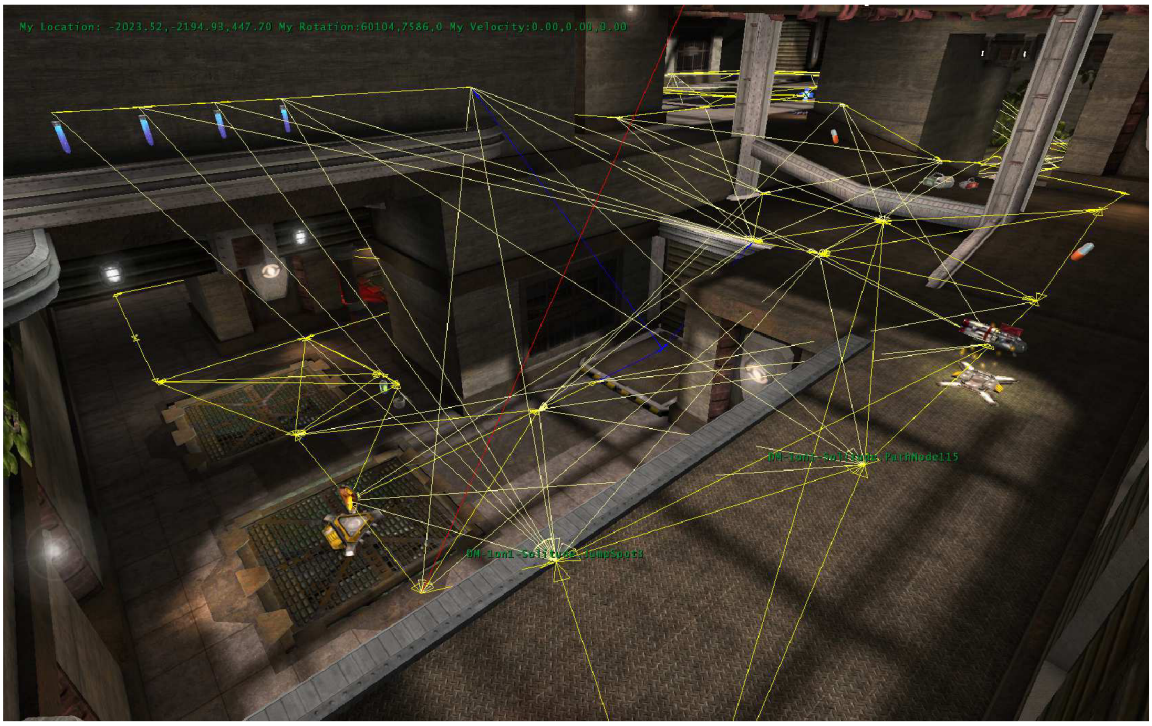
Situace je ale složitější, někdy je potřeba, aby agent provedl speciální akci k dosažení cíle, jako třeba použití výtahu, nebo přeskočení propasti na mapě. Z toho důvodu UT2004 obsahuje různé typy navigačních bodů a hran s informací, co je potřeba udělat, k úspěšné cestě po hraně, nebo zda je bod něčím speciální. Nejčastější typy bodů:

- `PathNode` – Bezpečný bod, kde se nic speciálního nevyskytuje. Nejčastější.
- `PlayerStart` – Stejné jako `PathNode` s tou výjimkou, že se může jednat o startovací pozici hráče.
- `InventorySpot` – Bod, kde se vyskytují předměty.
- `LiftCenter` – Střed výtahu. Pozice výtahu není statická.
- `LiftExit` – Body spojené s `LiftCenter` hranou. Možné východy z výtahu.
- `JumpSpot` – Bod, ze kterého je potřeba vyskočit, nebo skočit na něj.
- `Teleport` – Bod ze kterého je hráč teleportován na jiný bod na mapě.

7.3 Události

Agentův pohled na svět je poskytován přes IWorldView rozhraní. Toto rozhraní slouží jako agentovy základní smysly a jednoduchá paměť. Prostředí rozděluje na:

- **Objekty** (`IWorldObejct`) – hráči, předměty, atd.
- **Události** (`IWorldEvent`) – agent uslyšel zvuk, narazil do zdi, atd. Jsou rozděleny do dvou kategorií:
  - **Události objektů** (`IWorldObjectEvent`) – existuje 5 událostí tohoto typu:
    - První setkání (`WorldObjectFirstEncounteredEvent`) – aktivována když se agent setkává s nějakým objektem poprvé.



Obrázek 7.1: Ukázka části navigačního grafu na mapě DM-1on1-Solitude

    - Objevení (`WorldObjectAppearedEvent`) – objekt vstoupil do agentova zorného pole
    - Aktualizace (`WorldObjectUpdatedEvent`) – stav objektu byl aktualizován
    - Zmizení (`WorldObjectFirstEncounteredEvent`) – objekt zmizel z agentova zorného pole
    - Zničení (`WorldObjectFirstEncounteredEvent`) – objekt byl zničen
- **Události nespojené s žádným objektem** (čistý `IWorldEvent`) – příkladem takové události může být `HearNoise` která se spustí, když bot uslyší nějaký zvuk, nebo `BotBumped`, která je posledná kdykoli agent do něčeho narazí.

Při implementaci bylo vytvořeno několik posluchačů událostí, které na vzniklou událost reagují. Příklad takového posluchače je ukázan na kódu 7.2. Ten vytváří `EventListener` pojmenovaný `hearNoise`. Ten se zavolá pokaždé, kdy agent uslyší nějaký zvuk a následně se může otočit směrem ke zdroji zvuku, nebo k němu i běžet.

```
@EventListener(eventClass = HearNoise.class)
public void hearNoise(HearNoise event) {
double noiseDistance = event.getDistance(); // 100 ~ 1 meter
Rotation faceRotation = event.getRotation(); // rotate bot to face the
    location of the noise
log.info("HEAR NOISE: distance = " + noiseDistance);
}
```

Ukázka kódu 7.2: Ukázka použití EventListeneru

Kapitola 8

Testování

Testování hrálo velmi důležitou roli v tvorbě agenta od samého počátku až po konečnou verzi. Byly použity dva základní způsoby testování, první je popsán v podkapitole 8.1, druhý v podkapitole 8.2. Celkové výsledky obou způsobů testování jsou pak popsány v podkapitolách 8.3 a 8.4 a to od prvotní až po konečnou verzi.

8.1 Průběžné testování

Testování probíhalo téměř neustále, při každé změně. Agenta nešlo programovat "na slepo", každá změna se musela vyzkoušet přímo ve hře, protože i když kód vypadal správně, agent se mohl v konečném výsledku chovat jinak, než se očekávalo. Sledování agenta v akci byl asi nejsilnější a nejužitečnější odlaďovací nástroj, protože šlo vidět, kde se přesně agent nachází, co dělá, zda se někde zaseknul, atd.

Samotné rozšíření Pogamutu, pak obsahovalo nějaké další vizualizační funkce, které se hodili pro pozorování agenta. Stisknutím `CRTL + H` v rozhraní hry, se zobrazí nápověda se všemi možnostmi zobrazení (lze zde například stisknutím `CRL + G` zobrazit navigační mřížku, nebo stisknutím `CRL + D` zobrazit debugovací informace přímo na obrazovce).

Při testování se využívalo i logovacích záznamů, což jsou mechanizmy pro sledování programu pomocí vypisování zpráv. To lze využít jak pro ladění, tak pro statistické vyhodnocování informací.

Při tvorbě algoritmů se pak vždy postavila starší verze bota proti verzi novější, aby se ověřilo, zda skutečně provedená změna chování vylepšila, nebo naopak zhoršila.

8.2 Testování proti nativním agentům hry

Toto testování je i 4. bodem zadání bakalářské práce. Agent se postavil proti nativním herním agentům samotné hry UT2004.

Dle [5] existuje 8 typů těchto nativních botů:

1. **Novice** – 60% rychlost pohybu, 30° zorné pole, při souboji se téměř nepohybuje, 30° rozptyl střelby.
2. **Average** – 70% rychlost, 30° zorné pole, lepší přesnost střelby.
3. **Experienced** – 80% rychlost, 40° zorné pole, může se pohybovat a střílet zároveň.

4. **Skilled** – 90% rychlost, 60° zorné pole, umí dvojitý skok.
5. **Adept** – plná rychlost, 80° zorné pole, snaží se uskakovat projektilům.
6. **Masterful** – plná rychlost, 100° zorné pole, umí používat obtížnější herní taktiky (shock combo, shield jump), nejbližší skutečnému člověku, lepší vlastnosti jsou už považovány za nelidské.
7. **Inhuman** – 120° zorné pole, bez vzdálenostního limitu dohlédnutí.
8. **Godlike** – 360° zorné pole, bez omezení (vidí celou mapu po celou dobu), pro běžné hráče téměř neporazitelný, GameBots2004 ho nepoužívá.

Testování probíhalo dle pravidel turnaje Pogamut Cup, na mapách DM-1on1-Roughinery-FPS, DM-1on1-Solitude a DM-1on1-Backspace, do získání deseti bodů, nebo uplynutí časového limitu 10 minut.

8.3 Testování počáteční verze

První spustitelná verze, měla reaktivní schopnosti (zpozorování nepřítele, střelba), jednoduchý algoritmus pohybu po mapě a výběru zbraně.

Pří průběžném testování vykazoval občasné chyby. Mezi hlavními třeba zaseknutí pohybu, konání zbytečných a nelogických pohybů. Občas prokazoval snahu vyskočit na místo, kam doskočit možné nebylo. Často i nebral blízké lékárničky, či náboje, i když je akutně potřeboval.

I přes to všechno však byl z větší části efektivní a neměl žádnou chybu, která by mu bránila v hratelnosti. Proto bylo první testování souboje s nativními boty provedeno již na něm. Výsledky jsou popsány v tabulce 8.1. Stručně řečeno dokázal i přes popsané nedostatky porazit první tři složitostní typy botů.

8.4 Testování konečná verze

Konečná verze měla implementovány již všechny navržené algoritmy a testováním byla zbavena všech zjištěných nedostatků.

Využívá také úskoky a to jak pro rychlejší pohyb po mapě, tak při souboji s nepřítelem, což z něj dělá hůře zasažitelný cíl.

Výsledky jsou popsány v tabulce 8.2.

Oproti prvotní verze si dost polepšil, nyní dokázal porazit prvních 6 typů nativních botů, na jedné z map dokonce i nejobtížnější typ.

Zápasy na mapě DM-1on1-Roughinery-FPS							
Obtížnostní typ	1	2	3	4	5	6	7
MyBot (zabití/úmrtí)	5/1	6/3	8/3	4/7	4/9	2/10	3/12
NatBot (zabití/úmrtí)	1/5	4/7	3/8	7/4	9/4	10/2	10/1
Celkový čas	10:00	10:00	10:00	10:00	10:00	5:27	9:36
Celkové skore	5 : 1	6 : 2	8 : 3	4 : 7	4 : 9	2 : 10	-1 : 10
Výherce	MyBot	MyBot	MyBot	NatBot	NatBot	NatBot	NatBot
Zápasy na mapě DM-1on1-Solitude							
Obtížnostní typ	1	2	3	4	5	6	7
MyBot (zabití/úmrtí)	10/2	10/1	10/4	7/10	4/9	1/10	1/11
NatBot (zabití/úmrtí)	2/10	2/11	4/10	10/7	9/4	10/1	10/0
Celkový čas	5:12	5:36	7:17	9:19	10:00	5:55	5:12
Celkové skore	10 : 2	10 : 0	10 : 4	7 : 10	4 : 9	1 : 10	-1 : 10
Výherce	MyBot	MyBot	MyBot	NatBot	NatBot	NatBot	NatBot
Zápasy na mapě DM-1on1-BackSpace							
Obtížnostní typ	1	2	3	4	5	6	7
MyBot (zabití/úmrtí)	10/1	3/3	7/4	6/10	1/10	2/11	0/10
NatBot (zabití/úmrtí)	1/10	4/4	4/7	10/6	10/1	10/1	10/0
Celkový čas	7:08	10:00	10:00	7:59	7:06	7:57	5:43
Celkové skore	10 : 1	3 : 2	5 : 2	6 : 10	1 : 10	-1 : 10	0 : 10
Výherce	MyBot	MyBot	MyBot	NatBot	NatBot	NatBot	NatBot
Celkový výherce	MyBot	MyBot	MyBot	NatBot	NatBot	NatBot	NatBot

Tabulka 8.1: Výsledky zápasů první verze agenta

Zápasy na mapě DM-1on1-Roughinery-FPS							
Obtížnostní typ	1	2	3	4	5	6	7
MyBot (zabití/úmrtí)	10/0	10/0	10/1	10/2	10/4	10/3	3/10
NatBot (zabití/úmrtí)	0/10	1/11	1/10	2/10	4/10	3/10	10/3
Celkový čas	5:43	6:39	7:49	8:13	8:36	9:27	9:01
Celkové skore	10 : 0	10 : -1	10 : 1	10 : 2	10 : 4	10 : 3	3 : 10
Výherce	MyBot	MyBot	MyBot	MyBot	MyBot	MyBot	NatBot
Zápasy na mapě DM-1on1-Solitude							
Obtížnostní typ	1	2	3	4	5	6	7
MyBot (zabití/úmrtí)	11/1	10/0	10/0	10/3	10/7	7/10	6/10
NatBot (zabití/úmrtí)	3/13	0/10	1/11	3/10	7/10	10/7	10/6
Celkový čas	6:29	5:40	5:18	6:36	6:42	6:37	7:02
Celkové skore	10 : -3	10 : 0	10 : -1	10 : 3	10 : 7	5 : 10	6 : 10
Výherce	MyBot	MyBot	MyBot	MyBot	MyBot	NatBot	NatBot
Zápasy na mapě DM-1on1-BackSpace							
Obtížnostní typ	1	2	3	4	5	6	7
MyBot (zabití/úmrtí)	10/0	10/0	10/0	10/1	10/1	10/3	10/5
NatBot (zabití/úmrtí)	0/10	0/10	1/11	1/10	1/10	3/10	5/10
Celkový čas	4:08	4:23	5:08	6:29	6:14	7:11	7:23
Celkové skore	10 : 0	10 : 0	10 : -1	10 : 1	10 : 1	10 : 3	10 : 5
Výherce	MyBot	MyBot	MyBot	MyBot	MyBot	MyBot	MyBot
Celkový výherce	MyBot	MyBot	MyBot	MyBot	MyBot	MyBot	NatBot

Tabulka 8.2: Výsledky zápasů konečné verze agenta

Kapitola 9

Závěr

Cílem této práce bylo navrhnout a naimplementovat herního agenta-gamebota pro hru Unreal Tournament 2004 na platformě Pogamut.

Agent byl navrhnut na základě analýzy hry, známých taktik a obecné teorie agentů a následně vytvořen v programovacím jazyce Java.

Výsledkem je funkční agent, který dokáže hrát hru Unreal Tournament 2004 specializovaný na mód Deathmatch. Dokáže však dobře hrát i ostatní netýmové módy hry (Last man standing, InstaGib a Team Area Master).

Výsledný agent byl otestován na nativních agentech hry a dle výsledků v předchozí kapitole dopadl úspěšně.

V budoucnu by však bylo možné dodělat různá vylepšení. Mezi nimi například schopnost učení, nebo schopnost komunikace a spolupráce více takovýchto agentů v jiných, týmových módech hry jako například Capture the flag, Assault, Bombing run, či Onslaught, kde je týmová spolupráce důležitá a rozhoduje o celkovém výsledku hry.

Literatura

[1] BeyondUnreal Liandri Archives: Unreal Tournament 2004 [online]. [cit. 2014-20-05], http://liandri.beyondunreal.com/Unreal_Tournament_2004.

[2] Planet Unreal Tournament [online]. [cit. 2014-20-05], http://planetunreal.gamespy.com/View.php?view=UT2004GameInfo.Detail&id=1&game=4.

[3] Pogamut Cup [online]. [cit. 2014-20-05], http://www.pogamutcup.com.

[4] *Unreal Tournament 2004 manual*.

[5] Unreal Wiki: Legacy:Bot Mind [online]. [cit. 2014-20-05], http://wiki.beyondunreal.com/Legacy:Bot_Mind#Skill_Levels.

[6] Unreal Wiki: UnrealScript [online]. [cit. 2014-20-05], http://wiki.beyondunreal.com/UnrealScript.

[7] Bída, M.: GameBots2004 user documentation and network protocol [online]. [cit. 2014-20-05], http://pogamut.cuni.cz/pogamut_files/latest/doc/gamebots/.

[8] Bida, M.; Cerny, M.; Gemrot, J.; aj.: *Herrlich, M., Malaka, R., Masuch, M. (eds.) ICEC 2012*, svazek 7522, kapitola Evolution of GameBots project. Springer, Heidelberg, 2012, s. 397–400, http://artemis.ms.mff.cuni.cz/main/papers/Evolution_Of_Gamebots-2012-07-03_camera.pdf.

[9] Gemrot, J.; Kadlec, R.; Bída, M.; aj.: *Agents for Games and Simulations*, kapitola Pogamut 3 Can Assist Developers in Building AI (Not Only) for Their Videogame Agents. LNCS 5920, Springer, 2009, s. 1–15, http://pogamut.cuni.cz/main/papers/pogamut3_AGS_final.pdf.

[10] Kubík, A.: *Inteligentní Agenty*, svazek 1. vydání. Brno: Computer Press, 2004, iSBN 80-251-0323-4.

[11] Pogamut-Devel: guidelines:bots_in_fps_games [online]. [cit. 2014-20-05], http://pogamut.cuni.cz/pogamut-devel/doku.php?id=guidelines:bots_in_fps_games.

[12] Pogamut-Devel: guidelines:ut2004_objects [online]. [cit. 2014-20-05], http://pogamut.cuni.cz/pogamut-devel/doku.php?id=guidelines:ut2004_objects.

[13] Scott, B.: *AI Game Programming Wisdom*, kapitola The Illusion of Intelligence. Charles River Media, 2002, s. 16–20.

[14] Wikipedia: Unreal tournament 2004 [online]. [cit. 2014-20-05], http://cs.wikipedia.org/wiki/Unreal_Tournament_2004.

[15] Zbořil, F.: *Plánování a komunikace v multiagentních systémech.* Dizertační práce, Brno: Fakulta informačních technologií VUT v Brně, 2004.

Příloha A

Obsah CD

Přiložené CD obsahuje následující soubory:

- `projekt.pdf` – Bakalářská práce ve formátu PDF
- `projekt-tisk.pdf` – Bakalářská práce ve formátu PDF určená pro tisk
- `projekt.zip` – Zdrojové soubory bakalářské práce
- `src.zip` – Zdrojové kódy výsledného agenta

Příloha B

Popis instalace a spuštění

Tato přiložená kapitola obsahuje návod, jak zprovoznit platformu Pogamut a výslednou bakalářskou práci.

1. Instalace potřebných souborů:
   - Java JDK 1.6 nebo 1.7
   - Netbeans / Eclipse
   - Unreal Tournament 2004 (možno nainstalovat i z kolekce Unreal Anthology)
     Digitální distribuce: http://www.gog.com/game/unreal_tournament_2004_ece
   - Pogamut UT2004 Installer 3.6.1
     Ke stažení zde: http://diana.ms.mff.cuni.cz/main/tiki-index.php?page=Download
2. Importování kódů agenta do vybraného vývojového prostředí
3. Spuštění GB2004 serveru přes příkazový řadek příkazem:

```
UT2004_HOME_DIRECTORY/System/ucc.exe server
DM-TrainingDay?game=GameBots2004.BotDeathMatch?timelimit=999999
```

Ukázka kódu B.1: Příkaz pro spuštění GB2004 serveru

   Kde "DM-TrainingDay"může být nahrazen jakoukoli jinou Deathmatch mapou.

4. Spuštění kódu agenta ve vybraném vývojovém prostředí