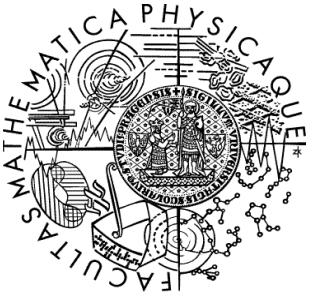
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Ondřej Burkert

Spolupracující dvojice do Unreal Tournamentu

Katedra software a výuky informatiky

Vedoucí bakalářské práce: Mgr. Cyril Brom

Studijní program: Informatika, Obecná informatika

2006

Rád bych na tomto místě poděkoval Cyrilovi Bromovi za vedení projektu a podnětné připomínky při psaní této práce. Dále bych rád poděkoval Jakubu Gemrotovi, který byl hlavním tvůrcem Pogamutu. Také bych chtěl poděkovat tvůrcům ostatním platforem, na nichž vznikl projekt, kterým se tato práce zabývá.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne Ondřej Burkert

Obsah

Název práce: Spolupracující dvojice do Unreal Tournamentu
Autor: Ondřej Burkert
Katedra: Katedra software a výuky informatiky
Vedoucí bakalářské práce: Mgr. Cyril Brom
e-mail vedoucího: brom@ksvi.mff.cuni.cz

Abstrakt: Práce je zaměřena na problematiku týmové spolupráce dvou agentů v prostředí hry Unreal Tournament (UT) v základním týmovém módu hry. Řešil jsem hlavně realizace autonomního chování, komunikace, synchronizace důležitých informací a implementace taktických operací. Vhodné využití lepší informovanosti agenta vede ke zlepšení jeho herních dovedností a uvěřitelnosti. Projekt byl implementován na platformě tvořené spojením hry UT, middleware Gamebots, architektury pro návrh logiky agentů POSH a middlewaru Pogamut. Po úspěšné implementaci jsem provedl sadu testů, kterými jsem zjišťoval dopad některých postupů na chování agentů. Zjistil jsem, že míra spolupráce ovlivňuje výsledný herní projev. Dvojice agentů, která spolupracuje jen na základě předávání informací, se ukázala jako velmi efektivní a vhodná pro daný mód hry.

Klíčová slova: týmová umělá inteligence, First-Person Shooters, bot, POSH

Title: Unreal Tournament Twins
Author: Ondřej Burkert
Department: Department of Software and Computer Science Education
Supervisor: Mgr. Cyril Brom
Supervisor's e-mail address: brom@ksvi.mff.cuni.cz

Abstract: This bachelor thesis is focused on team cooperation of a couple of agents in the enviroment of Unreal Tournament (UT) in the fundamental team-play mode. I worked on realization of autonomous behavior, communication, synchronization of important informations and implementation of tactical operations. Better informed agent could deliver more sofisticated performance and credibility. I implemented the project on the framework made from the game UT, the middleware Gamebots, POSH (the architecture for making agent's logic) and the middleware Pogamut. When I succesfully finished implementation, I have made set of tests to find out the impact of some approaches on agents performance in the game. I found out that the level of cooperation affects the performance. Couple of agents who were cooperating only by sharing informations proved to be the most effective and suitable for the choosen mode of the game.

Keywords: team artificial inteligence, First-Person Shooter, bot, POSH

Kapitola 1

Úvod

Hlavním cílem projektu, kterým se zabývá tato práce, byly návrh a implementace spolupracující dvojice do hry *Unreal Tournament* [1] (dále UT), mód *Team Deathmatch* (dále TDM). UT patří mezi hry typu *First-Person Shooter* (dále FPS). Tyto hry se odehrávají v reálném čase v 3D simulaci reálného světa. Hráči vnímají dění z pohledu postavy ve hře. Mezi základní herní činnosti patří eliminace protivníků. Tento typ her je typicky založen na podpoře her několika hráčů připojených k hernímu serveru, a proto jsou dnes velmi populární.

Jak již bylo zmíněno, her se účastní postavy. Obecné postavě ve hře ovládané člověkem či počítačem budu říkat *avatar*. Pro počítačem ovládaného avatara se vžilo označení *bot*. Boti reprezentují lidské, humanoidní či jiné protivníky. Hráč proto očekává, že se budou patřičně (např. lidsky) chovat. Návrh a implementace takového chování je proto složitá.

Cílem projektu bylo navrhnout a implementovat boty, kteří budou schopni v prostředí UT správně fungovat, plnit cíle hry a navíc navzájem spolupracovat. Na počátku stál bot v prostředí hry (dále *mapa* nebo *lokace*), vnímal okolí, ale nijak nereagoval. Bylo třeba vyřešit autonomní i týmové chování botů. Rozhodovací a plánovací algoritmus, který ovládá bota, budu označovat heslem *logika bota*.

Při realizaci projektu jsem použil několik platforem. Na zjednodušení ovládání bota v UT jsem použil *Gamebots* [2] (dále GB). GB je middleware vyvinutý skupinou studentů na University of Southern California, který umožňuje připojit do UT externí boty. Pracuje na principu klient – server. Externí logika dostává informace o tom, co bot vnímá, a posílá příkazy, které bot vykonává.

Pro návrh logiky využívám architekturu *Parallel-rooted, Ordered Slip-stack Hierarchy* [3] (dále POSH) a *Behavior Oriented Design* [4] (dále BOD). BOD je metodologie, která poskytuje návod k dekompozici složitého cílového chování na jednodušší části. POSH je poměrně nová architektura, která byla vyvinuta na základě BOD Joannou J. Bryson [5] na University of Bath. Slouží k tvorbě a spouštění reaktivních plánů (hierarchicky uspořádaná if – then pravidla), které rozhodují o činnosti bota.

Pro spouštění a ladění botů používám Pogamut [6], což je aplikace propojující UT, Gamebots a implementaci POSHe v Pythonu – pyPOSH [7].

Při návrhu týmové spolupráce jsem se inspiroval článkem van der Sterrena [8]. Boti spolu komunikují pomocí zpráv posílaných skrze komunikační kanál v UT. Takto získané informace zohledňují při výběru následující akce. Díky tomu se mohou držet pohromadě, nepřekážet si ve střelbě, systematicky se ozbrojovat a organizovaně útočit. Velmi důležitým prvkem se ukázalo také kvalitní provedení autonomního chování samotného bota, které má významný vliv na výsledné chování botů.

Na nastíněném spojení platforem pracujeme na MFF UK tři. Jakub Gemrot pracuje na projektu NavMesh, který by měl zpřístupnit botům pokročilé vnímání mapy. Michal Bída vyvinul Emoční boty. Představují pokus o implementaci emočního modelu a projevů emocí v UT. Michal Bída řeší, podobně jako já,

problémy související s autonomním chováním bota. Naše projekty se liší řešením těchto problémů a nadstavbou nad autonomním botem. Zatímco Michal Bída se zabýval projevem emocí, já jsem se zaměřil na týmovou spolupráci.

Spojení platforem bylo popsáno v článku [9]. Při popisu méně podstatných částí práce (popis platforem) jsem s úpravami přejímal některé pasáže.

V následující kapitole vytyčím hlavní cíle projektu. V kapitole 3 popíšu použité platformy a následně v kapitole 4 použitou metodologii. Kapitolu 5 naplňuje podrobnější popis problémů souvisejících s cíli projektu a jejich řešení. Následující kapitola je věnována testům, kterými jsem zjišťoval dopad různých úrovní spolupráce na projev botů. V kapitole 7 jsou popsány práce související s tématem projektu. Další kapitola popisuje možné další práce v této oblasti. V 9 kapitole následuje celkové shrnutí přínosů práce. Po výčtu použité literatury popíši obsah přiloženého CD.

Kapitola 2

Cíle projektu

Na počátku byly pouze aplikace zmíněné v úvodu a bot stojící nečinně v lokaci.

Hlavní cíl projektu. Hlavním cílem projektu bylo navrhnout a implementovat spolupracující dvojici do módu hry TDM. Tento cíl jsem si rozdělil na několik podcílů. Jsou to:

- komunikace
- společný pohyb
- reakce na nepřítele
- pohyb po mapě
- sbírání předmětů

Realizaci jednotlivých podcílů podrobně rozeberu později. V následujících dvou kapitolách představím platformy, na nichž projekt vznikal, a použitou metodologii. Tyto měly významný vliv na návrh a implementaci řešení jednotlivých problémů.

Související cíle projektu. Mezi související cíle projektu patřilo dosáhnutí takové úrovně vytvořených botů, aby byli kvalitativně srovnatelní s originálními boty v UT. Dalším cílem byl pokus o implementaci spolupracující dvojice tak, aby bylo možné vytvořit dvojice s různými úrovněmi spolupráce pouhou modifikací plánu. Což se povedlo a umožnilo to provést zajímavější testování použitých přístupů.

Kapitola 3

Použité platformy

Na následujícím obrázku je názorně ukázáno, jak jsou jednotlivé platformy propojeny. V této kapitole se budu věnovat popisu jednotlivých částí. První se zaměřím na obecnou charakterizaci FPS her, dále budou následovat pasáže o UT, GB, pyPOSHi a Pogamutu.

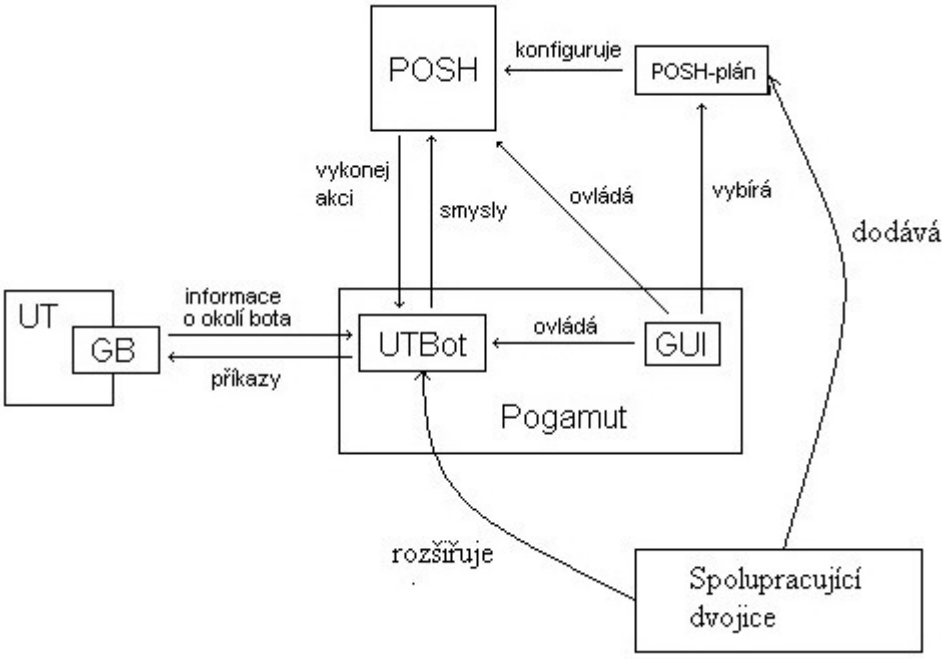


Fig. 1. Znázornění provázání jednotlivých aplikací výsledného spojení a zapojení projektu.

3.1 Hry typu First-Person Shooter

Hry typu FPS se odehrávají v reálném čase a v simulaci reálného světa. Hráči se pohybují prostředím a plní úkoly dané módem hry. Úkoly většinou vyžadují eliminaci protivníka pomocí dostupných zbraní. FPS hry podporují hru několika hráčů připojených k hernímu serveru, a proto jsou dnes velmi populární.

V komerčních herních titulech byl v minulosti kladen značný důraz na vývoj grafických elementů. Grafická stránka hry, engine hry a další součásti dostávaly a stále ještě dostávají významnou část výkonu stroje na němž běží (například přibližně 90% výkonu CPU). Důsledkem je, že hry jsou dnes po grafické stránce srovnatelné. Prodejnost a hratelnost her tak začíná výrazně ovlivňovat i dobré zpracování umělé inteligence.

Umělou inteligenci v FPS hrách reprezentují boti. Boti zastupují lidské, humanoidní či jiné protivníky. Hráči pak očekávají, že se budou patřičně chovat. Navrh a implementace takové umělé inteligence představují komplexní úlohu.

Rychlé tempo hry zahlcuje hráče mnoha vjemy. Situace bota je podobná a jeho logika musí tyto vjemy konzistentně a korektně zpracovat a reagovat na ně. Výsledkem by pak mělo být chování, které rozumným způsobem pokrývá všechny situace. Bot by neměl zůstat někde bezradně stát, měl by si všímat protivníků a spolupracovat se spoluhráči. Neměl by také nestřílet absolutně přesně nebo například vnímat dění za sebou (nepůsobilo by to moc věrohodně).

Chování bota by mělo připomínat chování člověka. Boti spolupracují, komunikují spolu, při kontaktu s nepřítelem k němu aktivně přistupují a zůstávají při boji v pohybu, aby se nestali snadnými terči palby. Projevem se tedy blíží běžnému hráči.

Snahy o dosažení takového projevu se dnes ubírají mnoha směry. Zkoumá se možnost použití emočních modelů, lepší reprezentace lokací i právě týmová spolupráce. Všechny snahy spojuje jedna motivace. Vytvořit protihráče, proti kterým je radost hrát. Hlavním cílem každé hry je bavit hráče.

Se zábavností hry souvisí dva problémy. Za prvé je to vyváženost umělé inteligence. Nikoho nebaví hrát proti botům, kteří jsou tak hloupí, že je porazí i v podroušeném stavu. Na druhou stranu není dobré, když boti nedají člověku šanci a nebo dokončí misi bez ohledu na jeho přispění. Najít správnou rovnováhu mezi těmito extrémy může být velmi náročné.

Druhým problémem je uvěřitelnost bota. Uvěřitelnost můžeme chápat jako míru podobnosti chování bota projevu lidského hráče. Už v samotné definici je několik drobných zádrhelů. Hráči nehrají striktně lidsky, resp. nejednají v dané situaci jako člověk, který by jí byl opravdu vystaven. Vrhají se sebevražedně do víru střelby, aby přinesli taktickou výhodu svému týmu, jednají sobecky nebo nespolupracují. Mají propracovanou strategii, ale nedrží se jí striktně, místy improvizují. Jinými slovy člověk je velmi komplikovaný a emulovat jeho chování je dost ne jednoznačný úkol.

Problémem výzkumu v této oblasti je, že komerční a akademická sféra spolu nespolupracují. Také ne všechny myšlenky jsou publikovány v anglickém jazyce, což může vést k několikerému zkoumání týchž problémů.

3.2 Unreal Tournament™

Unreal Tournament byl vydán v roce 1999 firmou Epic Megagames a mezi počítačovými hráči si rychle získal popularitu. Zaměřím se však na popis světa, který UT simuluje, a na praktické záležitosti související s tvorbou botů v tomto prostředí. Tato část je převzata s úpravami z článku [9].

Fig. 2. Pohled postavy na dění ve hře.

Prostředí Unreal Tournamentu.
Jak již bylo řečeno, UT patří mezi FPS hry. Každá hra se odehrává v nějakém prostředí – lokaci. UT obsahuje desítky různých typů lokací. Her se účastní avataři. Každý avatar má tyto základní údaje, které se v průběhu hry mohou měnit:

- Výdrž („health“), která udává kolik poškození je ještě daný avatar schopný snést než zemře.
- Brnění („armor“), které ovlivňuje míru poškození, které avatarovi způsobují zbraně ve hře.
- Zbraň („weapon“), kterou avatar právě drží (jeden avatar může vlastnit více zbraní, držet v jednu chvíli však může jen jednu z nich).

Další vlastnosti jako například rychlost pohybu či rozhled jsou pro všechny avatary stejné.

Navigační body. Součástí lokací jsou navigační body, tzv. *Navigation Points* (dále NP), které jsou standardně viditelné pouze pro boty. NP rovnoměrně pokrývají mapu a určují, kam se bot může nějakým způsobem dostat. Boti si mohou zjistit, zda se lze přímým pohybem po zemi dostat z jejich pozice ke zvolenému NP. Takto se dá vytvořit orientovaný graf, kde NP jsou uzly grafu a hrana mezi NP x a y existuje, pokud se lze dostat přímo z x do y. Tento graf pak botům slouží jako reprezentace lokace. Tedy boti vnímají mapu i své aktuální okolí jako množinu NP, a proto se nemohou pohybovat prostředím jako avatar ovládaný člověkem. Jednoduše nemají prostorovou představu o svém okolí.

Mapy. Pro boty je mapa množinou NP. Těch však existuje několik druhů a některé nesou dodatečné informace, které mohou boti využívat. Nejdůležitější jsou NP, *Spawn Pointy* (dále SP) a *Inventory Spoty* (dále IS). SP označují místa, kde se boti „respawnují“ neboli znovu objevují na mapě po své smrti. IS pak značí, kde se objevují zbraně, munice, lékárničky, brnění a jiné speciální předměty (neviditelnost, antigravitační boty apod.), které mohou boti sbírat a používat. Předměty se objevují stále na stejných místech. Dále pro usnadnění navigace bota existují *Jump Pointy*, které označují, kde má bot vyskočit. Také existují body označující vhodná místa pro přepadení, obranu, použití odstřelovací pušky apod. Mezi obvyklé prvky mapy patří

movery[1]. I ty jsou označeny speciálními navigačními body, které nesou informace o stavu zařízení a způsobu ovládání.

Boti v UT. UT boti jsou naprogramováni v UnrealScriptu[2]. Logika botů je založena na hierarchických konečných automatech. Chování je řízeno událostmi, které jsou vyvolány zprávami, jež jsou zasílány logice serverem hry. Pro každou zprávu je v každém stavu automatu definována metoda, která se vykoná po obdržení dané zprávy. Jedná se tedy o událostmi řízený model, kdy událost způsobuje změnu stavu automatu.

Zmíněného modelu využívá Gamebots, ale o tom až později.

Módy hry. UT obsahuje několik módů hry. Každý mód nějakým způsobem modifikuje pravidla a cíle hry. Módy jsou *Deathmatch*, *Team Deathmatch*, *Last Man Standing*, *Assault*, *Domination* a *Capture the Flag*. S výjimkou módu Las Man Standing platí, že pokud je avatar zabit soupeřem, objeví se znovu na nějakém SP v lokaci se základní konfigurací zbraní, výdrže a brnění.

V módu *Deathmatch* hrají všichni proti všem (boti i hráči) a úkolem je zlikvidovat co nejvíce soupeřů a být přitom sám likvidován co nejméně.

Team Deathmatch se od módu Deathmatch liší tím, že hráči hrají proti sobě v týmech, cílem je pak získat co nejvíce bodů za zabití nepřítele v rámci týmu.

Last Man Standing je opět podobný typu hry Deathmatch. Všichni hrají proti všem. Nicméně v tomto typu hry se hráči po smrti znova neobjevují a vyhrává hráč, který zůstane ve hře jako poslední.

Mód hry *Assault* rozděluje všechny hrající hráče na dva týmy. První tým útočí a druhý brání. Útočící tým má v časovém limitu splnit na mapě určitý úkol, např. dobýt pevnost, a bránící tým mu brání. Hodnotí se také rychlost provedení úkolu.

V módu hry *Domination* jsou na mapě umístěny tzv. *kontrolní místa*. Hráči jsou opět rozděleni na dva týmy. Vyhrává tým, který jako první dosáhne určité hranice bodů. Body se zvyšují každému týmu podle počtu obsazených kontrolních míst. Přičítání bodů probíhá přibližně každé 3 vteřiny.

V *Capture the Flag* (dále CTF) proti sobě stojí opět dva týmy. Každý tým má svou základnu a v ní umístěnou vlajku. Cílem týmů je ubránit svou vlajku a mezitím ukořistit vlajku protivníka.

Týmová inteligence. Každý z těchto módů klade různé nároky na logiku botů. V módu Deathmatch maximalizujeme počet zabitých protivníků. V Last Man Standing musíme přežít. Zde hrají boti každý sám za sebe, takže není týmová inteligence použita. Ostatní módy však kladou důraz na týmovou spolupráci. V Assault máme za úkol nalézt slabá místa v obraně soupeře při útoku nebo pokryt tato místa při obraně, takže je vhodné bránit či útočit koordinovaně. Domination klade důraz na optimální pokrytí prostoru vhodným zapojením všech členů týmu a mód CTF klade důraz na

1 *Mover* – pohybující se stěna, trampolína, výtah a podobně. Pro termín bohužel neexistuje vyhovující česká terminologie.

2 *UnrealScript* - speciální vnitřní jazyk UT. Existuje k němu editor, který umožňuje doplňovat existující kód a tvořit nové objekty ve hře.

plnění obranných i útočných úkolů a rozvržení týmové strategie. Otázkou je, jak je to v módu Team Deathmatch a zda i zde má smysl zapojit týmovou inteligenci.

Týmová inteligence originálních UT botů je založena na jednoduchém centralizovaném principu. V týmu existuje velitel a ten udílí příkazy. Boti si například umí zavolat o pomoc, pokud se dostanou do kontaktu s nepřítelem. Tento model není použit u módu Team Deathmatch. Zde boti jednají na vlastní pěst stejně jako v módu Deathmatch (samozřejmě nestřílejí po vlastních členech týmu).

3.3 Gamebots[3]

Gamebots [2] je klient-server aplikace napsána v UnrealScriptu, která slouží k ovládání avatara v UT. GB je server, ke kterému je možné připojit externí boty pomocí TCP/IP protokolu (fig. 3). GB poskytuje rozhraní pro vytvoření nového bota v UT a jeho další ovládání. Rovněž definuje textový protokol, pomocí něhož lze botovi dávat příkazy a ze kterého se dozvídáme informace o okolí bota. GB zasílá botovi pouze informace o tom, co právě vidí nebo slyší. Bot tedy vnímá reálné podněty, např. neví, jestli se někdo schovává za rohem. Také nemá žádnou paměť.

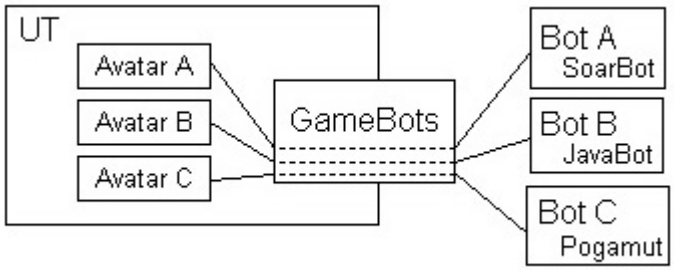


Fig. 3. GB spojuje avatara s různými logikami, které mohou být napsány v různých jazycích

GB zprávy. Mezi klienty a serverem GB probíhají tři typy zpráv:

- synchronní – od serveru ke klientovi; zprávy přicházejí v dávkách s frekvencí 10Hz a obsahují informace o viditelných navpointech, avatarech, předmětech, apod.
- asynchronní – od serveru ke klientovi; zprávy přicházejí vždy, když nastane méně obvyklá situace jako je zpráva od jiného hráče, smrt hráče, naražení do stěny, zvuk apod.
- příkazy – od klienta k serveru, příkazy ovládající bota

Výhody a nevýhody. Výhodou je zvolený model klient-server, který umožňuje napsání logiky bota v jakémkoli jazyce (obr. 4) a který obsahuje podporu síťové komunikace. Tedy umožňuje spouštět logiku bota na jiném počítači než server UT. GB velmi usnadňují vytvoření a ovládání agenta. Vývojář si nemusí osvojovat Unreal Script a přesto může využít hotové prostředí UT pro vývoj UI.

Mezi nevýhody patří maximální délka zprávy, kterou můžeme najednou poslat (1023 znaků). Což možná ústí v občasnou nestabilitu GB. Dále mají GB problém s

3 Tato pasáž byla přejata úpravou pasáže v [9]

korektním přístupem ke skokům a moverům. Ovšem pokud se omezíme na mapy, ve kterých není nutné příliš skákat a pohybovat se pomocí moverů, není to problém. Bot také nevnímá letící projektily (některé letí pomalu, takže se jim dá vyhnout). Sice existuje zpráva z GB, která tuto situaci ošetřuje, ale ve výsledku si bot ničeho nevšimne.

Asi nejzávažnější chybou je, že se někdy bot spuštěný skrze Gamebots bez zjevného důvodu odpojí od herního serveru. Tuto chybu se doposud nepodařilo lokalizovat a odstranit (viz [10]). Možné je, že se jedná o chybu UT, neboť UT boti se občas také odpojí ze hry a jsou nahrazeni automaticky dalšími boty. Dále občas UT, pravděpodobně vinou Gamebots, havaruje. A na závěr, neesteticky působí nedokonalé spouštění a ukončování animací, které místy ústí v nelogickou vypadající vizualizaci projevu.

Přesto jsou Gamebots vhodnou aplikací, neboť výrazné zjednodušení tvorby logiky bota převažuje nad zmíněnými nedostatky.

3.4 PyPOSH

PyPOSH je název implementace POSHe v Pythonu. Podrobný popis této aplikace je soušástí diplomové práce Andyho Kwonga [11]. Jedná se de facto o přepis původní implementace POSHe od J. J. Bryson [5] v Lispu do Pythonu. PyPOSH byl experimentálně napojen právě na Gamebots a tedy i UT. Ladící prostředí je poměrně strohé a nepřehledné. Neposkytuje komplexní podporu pro vývoj botů v UT, a proto byla vyvinuta aplikace Pogamut.

3.5 Pogamut

Pogamut je framework, který usnadňuje vývoj botů nad Gamebots a UT. Navíc volitelně můžeme použít pro vytváření logiky bota právě výše zmíněný PyPOSH. Pogamut byl vyvinut mým kolegou Jakubem Gemrotem [6] ve spolupráci se mnou a Michalem Bídou [12]. Skládá se ze dvou částí.

GUI. Slouží ke spouštění vlastních botů – lze spustit i několik botů najednou. Před spuštěním máme možnost upravit počáteční parametry jako IP počítače, na němž běží server hry, či proměnné, které si k botovi sami přidáme. Hlavní okno aplikace pak umožňuje sledovat žurnál GB zpráv, výběr akcí v logice bota a stav bota (pozice, zbraň, rychlost, zdraví). Dále můžeme zadávat příkazy a měnit nastavení proměnných za běhu. Také lze agenta zastavit a znovu spustit, nebo jej nevratně zabít.

Hlavní výhodou je, že GUI přehledně v jednom okně zachycuje vše, co potřebujeme o botovi vědět, což výrazně zjednodušuje ladění. Též není problém spustit dva typy botů proti sobě a sledovat, co dělají.

Bot.py Druhou částí aplikace je třída bot.py, která zastřešuje komunikaci s GB a vytváří tak mezivrstvu mezi GB a logikou bota. Obsahuje metody pro zpracování zpráv a zasílání příkazů. Také je zde řešena paměť bota i s mazáním příliš starých informací. Paměť obsahuje veškeré informace získané z přijatých zpráv od GB serveru.

Výhody a nevýhody. Pogamut velmi zjednodušuje vývoj logiky bota nad UT. Zahrnuje většinu obecně užitelných metod a datových struktur. Ty řeší problémy, s nimiž by se musel vypořádat každý, kdo by rád UI nad Gamebots a UT vyvíjel. Na dalším rozšiřování této aplikace se stále pracuje. Mezi rozšíření bude patřit například vytvoření profilů pro sledování zvolených zpráv či proměnných. Asi největším krokem bude integrace projektu NavMesh [6], který zpřístupní nové, pokročilé vnímání mapy pro boty.

3.6 Proč právě toto spojení platforem?

Mezi hlavní důvody patří hotový svět UT. Jsou zde jasné cíle a ucelená množina vjemů. UT také umožňuje snadnou vizualizaci chování bota. Navíc v UT nalezneme originální boty, které můžeme použít pro srovnání s vlastními.

Dalším důvodem je výrazné zjednodušení ovládání bota použitím GB. POSH poskytuje algoritmus vykonávající rozhodování. Posledním důvodem je existence aplikace Pogamut, která poskytuje přehled o dění v rámci GB a POSHe a tím usnadňuje ladění botů.

Kapitola 4

Metodologie

Nyní se zaměřím na použitou metodologii. První se budu věnovat metodě *Behavior Oriented Design* (dále BOD), následně popíši POSH, který ji rozšiřuje a aplikuje v praxi. Kapitolu uzavřu přehledem přístupů k návrhu týmové spolupráce.

4.1 Behavior Oriented Design

Behavior oriented design [4] můžeme přeložit jako design orientovaný na chování. Je to metoda, která definuje postup pro dekompozici složitého chování na jednodušší části. Slovo chování označuje dva různé termíny, proto budu používat indexů pro rozlišení. Chování A bude představovat celkové chování agenta. Chování B bude označovat modul, který je zodpovědný za podmnožinu chování A.

BOD je metodologie založená na objektově orientovaném programování a modulech chování. Komplexní chování A rozložíme na *činnosti* a *smysly* (paralela na chování člověka). Činnosti a smysly představují jednoduchá primitiva pro ovládání agenta. Smysly souvisejí s percepcí. Činnosti jsou zodpovědné za provádění akcí. Činnosti a smysly se sdružují do modulů zastoupených třídami – *chováními* (chování B). Chování B se pak stará o všechno spadající do jeho kompetence. To zahrnuje zpracování vjemů, zapouzdření souvisejících proměnných a správu příslušné paměti.

Postup při vytváření složitého komplexu chování A je iterativní [13]. Na začátku provedeme následující:

1. Specifikujeme, co chceme, aby agent dělal.
2. Popíšeme pravděpodobné aktivity agenta jako posloupnosti akcí.
3. Pro tyto akce definujeme příslušné činnosti a smysly.
4. Shromáždíme příbuzné elementy, budou tvořit základ knihovny chování (soubor modulů chování B).
5. Rozmyslíme si priority jednotlivých akcí (nutné pro tvorbu POSH plánu).
6. Vybereme první chování k implementaci.

Dále postupně implementujeme jednotlivé smysly a činnosti v daném modulu chování a vždy testujeme, zda je implementace korektní. Po implementaci všech rozumně specifikovaných části provedeme revizi specifikace. Tento postup opakujeme, dokud nedostaneme bota s požadovanými vlastnostmi.

BOD jsem použil, neboť je na něm postaven POSH. Navíc je tento přístup velmi intuitivní a finální implementace rozloženého chování A je jednodušší.

4.2 POSH

POSH („Parallel-rooted, Ordered Slip-stack Hierarchy“) představuje architekturu pro tvorbu logiky autonomních agentů, tedy i botů. Byl navržen a implementován v

Lispu Joannou J. Bryson [5] z University of Bath a posléze A. Kwongem [11] přepsán do Pythonu a zkušebně propojen s UT. Jak už bylo zmíněno, tato implementace se nazývá pyPOSH.

POSH je založen na rozšíření BOD. BOD definuje jednotlivé moduly chování a interpret plánů jim předává řízení dle zvoleného pravidla. Jednotlivé moduly mohou pracovat i paralelně.

Základem architektury jsou *POSH plány*. Představují zápis stromově uspořádaných „if-then“ pravidel. Každé pravidlo má podmínky a důsledky. Podmínka je konjunkce smyslů nebo jeden smysl. Důsledkem pravidla může být sekvence činností nebo množina pravidel a tedy další rozhodování. Právě toto uspořádání vytváří hierarchii pravidel. Výběr následující činnosti začíná u kořene hierarchie, kde jsou podle priority seřazena pravidla. Pravidla v kořeni se periodicky ověřují a řízení se předává prvnímu pravidlu se splněnými podmínkami. Jeho důsledkem pak je buď vykonání sekvence činností (třeba i jedné) nebo přechod na další úrovně rozhodování a následné provedení činnosti. Vykonávání činnosti, sekvence, rozhodování může být přerušeno při aktivaci pravidla v kořeni s vyšší prioritou.

```
((AP ap-behej (seconds 10) (behej))
 (C c-turn-all-around (seconds 3) (goal((turned-around)))
  (elements
    ((stop (trigger ((see-enemy))) finished-turn))
    ((turn (trigger ((not-see-enemy))) turn-left))
 ) )
(RDC life (goal ((fail)))
 (drives
    ((respond-to-damage (trigger ((taken-damage) (not-see-enemy))) c-look-
      all-around))
    ((run-somewhere (trigger ((succeed))) ap-behej))
) ) )
```

Fig. 4. Příklad POSH plánu, jehož interpretaci uvedu dále v textu. *AP* slouží pro zápis sekvence činností, *C* pro definici množiny rozhodovacích pravidel. *RDC* představuje kořen rozhodovacího stromu. Zápis pravidla je ve tvaru: ((jméno pravidla (podmínky spuštění (část *trigger*)) důsledek)).

Na obrázku (fig. 4) vidíme příklad jednoduchého plánu. Bot, který by byl takovým plánem řízen, bude náhodně procházet lokaci a pokud jej někdo zraní a nevidí nepřítele, otočí se dokola. K náhodnému procházení slouží pravidlo *run-somewhere*. Jeho důsledkem je sekvence *ap-behej*, která má jedinou činnost – *behej*. Pravidlo je úplně na dně kořene (*RDC life*) a jeho podmínka (*succeed*) je úspěšná vždy, takže se provádí dokud není vyrušeno vyšším pravidlem (*respond-to-damage*). To se spustí, pokud bota někdo zraní (*taken-damage*) a předá řízení dalšímu rozhodování (*c-look-all-around*). Je rozhodování podobné tomu v kořeni a bot se snaží otočit dokola (po 90°) a pokud zahlédne nepřítele, přeruší otáčení.

Výhody a nevýhody. Výhodou architektury je snadný zápis cíleného chování a snadná modifikace chování. To umožňuje vytvářet různé mutace vytvářeného chování, aniž bychom měnili knihovnu chování. Také lze úpravami zjistit, jaká posloupnost akcí je vhodnější. A především existuje interpret POSH plánů, který podle nich řídí agenta. Právě díky těmto výhodám jsem použil POSH pro návrh

logiky botů.

Nevýhodou je, že gramatika pro psaní POSH plánů neumožňuje předávat volaným metodám modulů chování proměnné. Nicméně jde porovnávat návratové hodnoty smyslů s čísly.

Míra dekompozice. Při dekomponování chování máme v zásadě tři možnosti.

Dekomponujeme co nejjemněji a výsledkem je poměrně velká množina smyslů a činností, které jsou však velmi prosté, a rozsáhlý POSH plán se složitou hierarchií pravidel. Tedy naprogramování smyslů a činností bude pravděpodobně jednoduché. Na druhou stranu může být příliš komplikovaný POSH plán zdrojem chyb a může se stát, že přestane být jasné, co agent dělá.

Pokud půjdeme opačnou cestou a provedeme hrubou dekompozici chování, získáme krátký a přehledný POSH plán. Ovšem smysly a činnosti budou místy velmi komplikované a někdy použitelné pouze pro jednu kontrétní situaci.

Například bot má mít smysl, který zjišťuje, zda má bojovat. Při jemné dekompozici jej bude tvořit konjunkce smyslů, které budou jednotlivě zjišťovat zda má zbraň, dostatečnou výdrž, viditelného nepřítele popřípadě další podmínky. Při hrubé dekompozici bude toto všechno sjednoceno do jednoho velkého smyslu, který bude použitelný jen pro aktivaci boje.

Zlatá střední cesta tedy bude asi tím nejvhodnějším řešením, a proto jsem se snažil dekomponování dělat rozumně jemně.

Volba stylu dekompozice by na výsledné chování neměla mít významný vliv.

4.3 Týmová inteligence – přístupy

Přístupy k návrhu týmové inteligence můžeme rozdělit dle [8] na centralizovaný a decentralizovaný.

Decentralizovaný přístup. Je založen na volném pojetí spolupráce. Boti spolu komunikují, předávají si informace o prostředí, protihráčích i o sobě. Tyto znalosti pak promítají do svého jednání. Ve skupině botů neexistuje velitel. Boti rozhodují každý sám za sebe. Tento přístup je velmi robustní a dobře se vyrovnává s širokou škálou situací. Zajímavé je, že se zde často vyskytují takzvané *emergentní jevy*. To jsou projevy botů, které nejsou součástí návrhu jejich logiky, ale nastávají v určitých situacích jako důsledek kombinace naprogramovaných reakcí a působí velmi promyšleným dojmem. Příklady emergentních jevů, které jsem zaznamenal, uvedu v kapitole, která je věnována realizaci projektu.

Ovšem tento přístup má i své nevýhody. Například těžko zvládá provádění dlouhodobých taktických operací. Tyto vyžadují přesné provádění příkazů a déle trvající spolupráci.

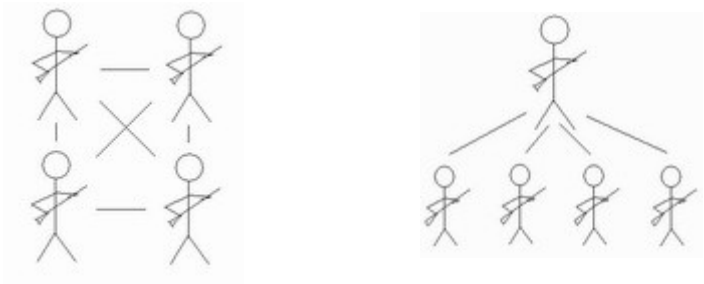


Fig. 5. Styl výměny informací a příkazů v decentralizovaném a centralizovaném přístupu.

Centralizovaný přístup. Je prakticky opakem předchozího. Ve skupině botů je delegován velitel, kterému ostatní podléhají. Velitel shromažďuje informace od svých podřízených a na jejich základě vybírá z množiny akcí, co bude jeho jednotka dělat, a udílí příslušné rozkazy. Přístup zvládá velmi dobře komplexní strategii. Problémy mohou nastat například po smrti velitele a přístup není tak flexibilní jako decentralizovaný přístup.

Hierarchický přístup. Vznikl rozšířením centralizovaného přístupu. Mezi spolupracujícími boty je vytvořena hierarchie hodností (inspirace armádou). Informace o prostředí a nepříteli plynou směrem vzhůru a podle důležitosti jsou buď poslány výš, nebo vyhodnoceny. Velitel (nejvyšší článek hierarchie) rozhoduje o strategii jednotky a deleguje úkoly svým důstojníkům. Ti mají pod sebou buď další menší jednotky řízené poddůstojníky, nebo už přímo vojáky a s jejich pomocí se snaží zadaný úkol splnit.

Trenér a dikrátor. Podle míry důvěry ve své podřízené rozlišujeme trenéra a diktátora (stále dle [8]). Trenér navrhuje vhodný postup a záleží na podřízených, jak se zařídí. Diktátor dává podřízeným přesné příkazy, které musí uposlechnout. Většinou se používá kombinace těchto postupů. Například boti dostávají příkazy s různou prioritou a podle ní se rozhodují stran plnění. V projektu používám právě kombinaci těchto přístupů. Boti reagují na příkazy podle jejich priorit, které jsou určeny prioritou v POSH plánu.

Volba přístupu závisí hlavně na aplikaci. Pro koordinaci početných jednotek se hodí víc hierarchický přístup. Pokud potřebujeme používat zhusta globálnější strategii, je vhodné použít centralizovaný přístup, který to dobře zvládá. Decentralizovaný přístup je vhodný pro volnou koordinaci agentů. Přístupy se dají mezi sebou kombinovat.

Kapitola 5

Spolupracující dvojice do Unreal Tournamentu

V této kapitole se budu podrobně věnovat realizaci projektu. Na začátku připomenu cíle projektu a nastíním strukturu projektu a návrh knihovny chování. Dále budu diskutovat použitý přístup k návrhu týmové spolupráce. Pak popíši řešení jednotlivých podcílů a kapitolu uzavřu náhledem na POSH plán bota. Všechny problémy jsem řešil pro případ co nejtěsněji spolupracující dvojice, další typy spolupráce jsem odvodil až na závěr modifikací plánu.

5.1 Cíle týmové spolupráce

Hlavním cílem bylo vyřešení týmové spolupráce, a proto se zaměřím hlavně na tuto část řešených problémů. V průběhu popíši i důležité části související s autonomním chováním bota. Pro připomenutí, spolupráce dvojice spočívá v těchto bodech:

- komunikace
- společný pohyb
- reakce na nepřítele
- pohyb po mapě
- sbírání předmětů

5.2 Struktura projektu

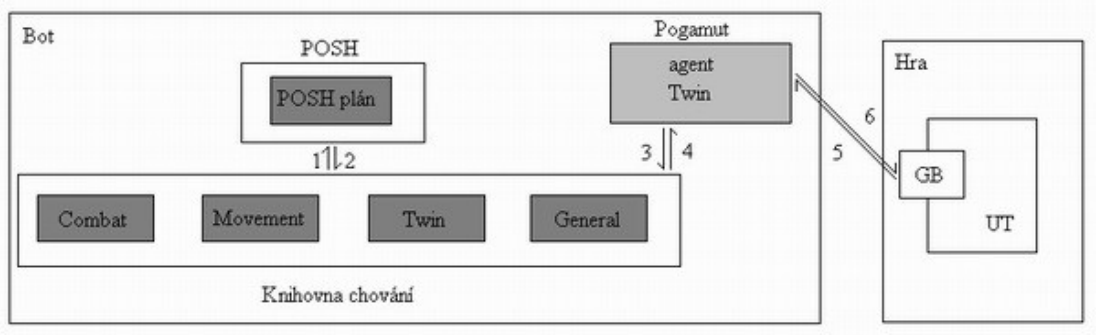


Fig. 6. Struktura projektu – bližší popis v textu.

Obrázek výše představuje hlavní části projektu a jejich vzájemné propojení. Tmavě vyznačeny jsou části, které jsem celé sám vytvořil, světleji část, kterou jsem převzal z Pogamutu a upravil pro potřeby aplikace, bíle části, do kterých jsem nijak nezasahoval. První popíšu jednotlivé části obrázku (Fig. 6). Základní rozdělení je na bota a hru, kterou tvoří GB a UT. Bot se skládá z Pogamutu a logiky bota, kterou tvoří POSH, resp. POSH plán, a knihovna chování obsahující jednotlivé moduly chování.

Nyní nastíním komunikaci mezi jednotlivými částmi:

1 Botovy smysly dávají odpovědi dotazům obsaženým v if-then pravidlech plánu.

2 Plán říká, která činnost se má vykonat, předává řízení danému modulu knihovny chování.

3 Smysly a činnosti získávají většinu informací z globální paměti bota, kterou spravuje třída Twin.

4 Činnosti posílají třídě Twin příkazy, co se má konkrétně provést.

5 GB server posílá botovi informace o jeho okolí, ty přijímá zpracovává třída Twin a ukládá je do paměti.

6 Třída Twin zajišťuje transkripci pokynů od činností do GB příkazů a posílá je serveru GB.

5.3 Knihovna chování

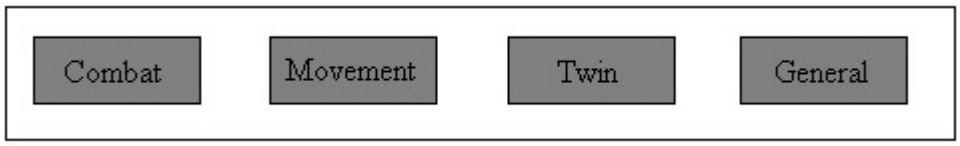


Fig. 7. Knihovna chování.

Pro návrh knihovny chování jsem použil BOD. Chování jsem rozdělil do čtyř tříd podle zaměření daných smyslů a akcí. Třídy jsou:

- pohyb (*Movement*) – toto chování se stará o vše související s pohybem a navigací bota. Patří sem tedy percepce předmětů a zvuků a akce, které slouží k navigaci bota k jakémukoliv typu věci nalézající se na mapě a pro pohyb po mapě
- boj (*Combat*)– toto chování obstarává botův projev v boji. Zahrnuje přístup k protivníkovi, střelbu, přepínání zbraní a smysly s tím související.
- obecné (*General*) – zde se nacházejí převážně smysly, které jsou spojeny se stavem bota.
- dvojice (*Twin*) – v tomto chování nalezneme vše související se spoluprací. Nacházejí se zde procedury zajišťující kompozici a posílání zpráv, příjem, zjišťování stavu spolubojovníka.

Bližší informace o jednotlivých třídách se nacházejí ve vývojové dokumentaci, která je součástí přílohy.

5.4 Týmová inteligence – diskuze řešení

V předchozí kapitole jsem představil několik možností, jak řešit týmovou spolupráci. V tomto projektu jsem použil decentralizovaný přístup. Domnívám se, že je pro Team Deathmatch (TDM) nejvhodnější. Důvody pro použití decentralizovaného přístupu jsou:

Struktura mapy. V módu hry TDM je hlavním cílem týmu zabít co nejvíc

protihráčů a zároveň se držet co nejdéle ve hře. *SpawnPointy* (SP) jsou rozmístěny rovnoměrně po celé lokaci a není zde žádný klíčový bod jako je tomu u modů Domination, CTF či Assault. Rozmístění SP pro nás znamená, že nelze předpokládat, jaká situace nastane po smrti bota a jeho následném objevení na mapě.

Taktické a strategické požadavky na UI. Globální strategie je v tomto módu nesmyslná. Oproti tomu použití taktických operací může být vhodné a může zvýšit uvěřitelnost botů. Navíc implementované taktické operace vystačí s volnou spoluprací. Pokročilejší vyžadují těsnější koordinaci, pak se však dá použít kombinace s centralizovaným přístupem.

Implementace jednoho POSH plánu. POSH plánu je použitelný pro oba boty. U centralizovaného plánu by byly potřeba minimálně plány dva (velitel, podřízený).

Emergentní jevy. Asi nejpodstatnější výhodou decentralizovaného přístupu je, že produkuje emergentní jevy. Například jeden bot narazí na nepřítele a informuje o tom svého spojence. Ten se může nacházet úplně jinde na mapě (třeba zrovna hledá zbraň). Na výzvu se vydá nejkratší cestou k nepříteli a tak se může stát, že mu vpadne do zad či do boku. Takže jej spolu sevřou do křížové palby a zvýší svou šanci na úspěch. Což vypadá velmi promyšleně, přestože to nebylo přímo naprogramováno. Díky tvarům map pro Team Deathmatch navíc nastávají poměrně často. U jiných módů jsou lokace koncipovány odlišně.

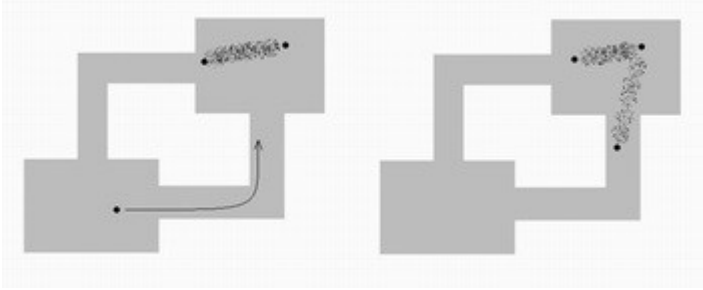


Fig. 8. Zjednodušené znázornění emergentního jevu. Bot byl zavolán na pomoc a díky tomu, že jde nejkratší cestou, se dostane nepříteli do boku.

5.5 Realizace podcílů projektu

Nyní podrobněji popíši, jak jsem realizoval jednotlivé podcíle projektu.

Komunikace. Komunikace je nezbytným základem pro jakoukoliv spolupráci. Pomáhá také zvýšit uvěřitelnost botů, neboť při smysluplné komunikaci z ní lze utvořit zprávy vnímatelné hráčem, které vypadají velmi věrohodně. Pak jsou potřeba dva komunikační kanály. Jeden pro přenos konkrétních informací a jeden pro propagaci zpráv k hráči. V UT máme soukromý a globální komunikační kanál, takže je možnost takový přenos emulovat. Nicméně pro ladění je vhodnější, když zprávy jdou přímo do globálního komunikačního kanálu.

Boti vnímají pouze zprávy od svého spojence, zbytek ignorují. Zprávy mají formát:

Jméno_bota Jméno_příkazu [Obsah_příkazu]

Příkazy jsou:

- INFO – slouží k předávání informací mezi boty, boti si předávají stav zdraví, zbraň, rychlost, pozici
- ENEMY – nese jméno a pozici nepřítele, kterého bot vidí
- ENGAGE – slouží k zahájení speciální operace *obestoupení*
- FOLLOW – spouští mód chování *následování*
- ERASE FOLLOW – slouží ke změně bota jdoucího na prvním místě
- EQUIPEMENT – přenáší informace o vybavení a jeho pozici

Nabízelo se také doplnit předávané informace o záměry botů. Bohužel jsou boti reaktivní. Jejich úmysly se neustále mění, a proto by zohledňování záměrů druhého bota pravděpodobně občas vedlo k přepínání mezi dvěma stavy (takzvaný *dithering*).

Společný pohyb. Informace o spolubojovníkovi získané z příkazu *INFO* umožňují botům pohybovat se společně. Tedy po objevení se na mapě se snaží hned po získání zbraně rychle najít druhého člena dvojice. Společný pohyb má několik výhod. První jde bot, který má vhodnější zbraň pro boj na blízko, víc životů, popřípadě brnění, takže se může lépe vypořádat s prvním náporem nepřítele. Druhý v pořadí se pravidelně otáčí a snižuje šanci na přepadení zezadu. Navíc mají zaručený lepší nebo stejný poměr sil při střetu s nepřítelem.

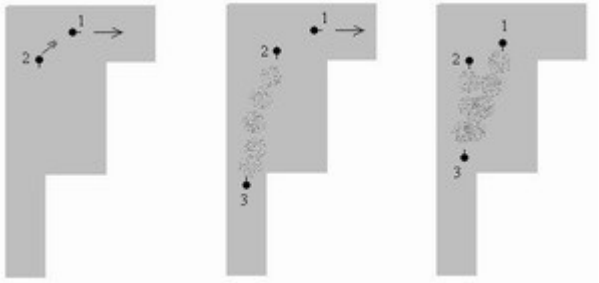


Fig. 9. Boti 1 a 2 spolupracují, bot 2 se právě ohlíží a uvidí nepřítele (3). Začne s ním bojovat a zároveň varuje bota 1. Ten se také začne věnovat nepříteli.

Ovšem není růže bez trnů a tento přístup má i své stinné stránky. Bot jdoucí první často překáží druhému ve střelbě. To je vyřešeno pomocí analytické geometrie. Boti mají informace o svých přibližných polohách. Druhý bot tedy jde na volnou pozici pro střelbu v závislosti na pozici nepřítele a prvního bota, který se také snaží ustoupit na stranu (Fig. 10). Přesto občas k zásahu spolubojovníka dochází.

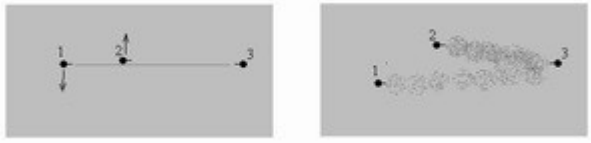


Fig. 10. Bot 2 překáží ve střelbě botovi 1. Oba jsou si navzájem vědomi svých pozic a tak každý ustoupí trochu na stranu a mohou bez obav útočit.

Dále je třeba občas změnit vedoucího bota (zraní se, druhý bot získá lepší zbraň

apod.). Tento problém jsem vyřešil příkazem *ERASE FOLLOW*, kterým si vynutí druhý bot v pořadí restart příslušných proměnných a následným *FOLLOW* změní vedení. Bez tohoto opatření by dvojice častěji prohrávala, neboť většinou dochází ke střetu ve směru pohybu a boti střílejí na nejbližšího bota. Když se boti střídají, může druhý bot přispět svou palebnou silou, i když už má hodně málo životů. Po boji mohou doplnit výdrž pomocí lékárniček a brnění a bojovat obstojně dál.

Reakce na nepřítele. Správná reakce na nepřítele je v FPS hrách klíčová. Proto jsem této oblasti věnoval zvýšenou pozornost. Při střetu s nepřítelem bot zahajuje palbu a úhybné manévry. Zároveň posílá zprávu o jeho pozici spolubojovníkovi. Během boje si udržuje informaci o pozici nepřítele v paměti. Takže pokud jej ztratí z dohledu, může běžet na místo, kde jej viděl naposledy a potencionálně jej najít a dorazit jej. To je velmi důležité. Pokud by si bot nepamatoval, že bojoval, začal by „řešit" něco jiného a třeba se k nepříteli otočil zády. Protivník by pak mohl vyjít zpoza překážky a zaútočil na zcela nepřipraveného bota.

Dále je velmi výhodné předat informaci o nepříteli svému spoluhráči pomocí příkazu *ENEMY*. Ten, pokud je sám tísněn soupeřem, zprávu ignoruje, ale pokud zrovna „má čas", snaží se dostat co nejkratší cestou na udanou pozici protivníka. Ne vždy se podaří přímo zasáhnout do boje, ale i napadení nepřítele oslabeného bojem může přinést kýžené ovoce. Když bot dorazí na poslední známé místo, kde viděl nepřítele, a nevidí ho, otočí se pro jistotu ještě dokola, zda se někde neschovává. Poté informace zapomene a pokračuje novým výběrem další činnosti.

Při boji samotném se boti snaží kličkovat před nepřítelem. Navíc se snaží k němu přiblížit nebo oddálit podle zbraně, kterou zrovna používají. Umí si také podle situace změnit zbraň, pokud mají vhodnou k dispozici. Tuto informaci Gamebots neposkytují, takže si bot musí pamatovat, které zbraně získal a udržovat si tento seznam aktuální.

Zvláštní případ nastává, pokud boti potkají nepřítele spolu. Pak se spouští příkazem *ENGAGE* jiný styl přístupu k nepříteli. Spočívá ve snaze obestoupit nepřítele, dostat jej do křížové palby. Tedy jeden z dvojice se snaží jít napravo a zároveň střílet po protivníkovi, druhý jde vlevo. Toto chování se pak vypne po smrti protivníka, nebo po smrti spolubojovníka. Výhodou tohoto jednání je také, že boti bojují společně, ale zároveň nejsou moc u sebe. Některé zbraně mají totiž i plošné ničivé účinky a je možné jimi zabít oba boty najednou. Dále tento přístup zvyšuje uvěřitelnost botů.

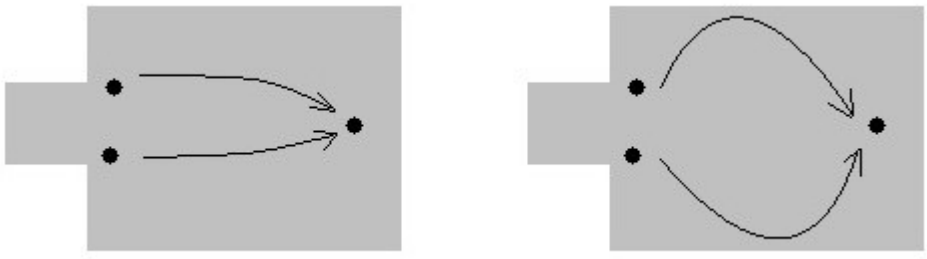


Fig. 11. Rozdíl mezi běžným přístupem k protivníkovi a přístupem za použití *ENGAGE*.

Pohyb po mapě. Důležité bylo také vyřešit pohyb botů po mapě, aby boti nezůstávali neustále v jedné části mapy. Řešení pomocí náhodného výběru z viditelných navigačních bodů či výběru nejméně navštíveného bodu se ukázalo jako příliš náhodné. Chování botů pak působilo velmi zmateně. Proto boti podobně jako UT boti obcházejí důležitá místa na mapě. V případě Team Deathmatch jsou to místa, kde se objevují zbraně či brnění, neboť jsou rozmístěny poměrně rovnoměrně po celé mapě a zároveň jich není příliš moc. Tento způsob obcházení mapy se aktivuje, když bot najde alespoň čtyři taková místa. Bot je pak, na rozdíl od UT botů, prochází v náhodném pořadí.

UT botům poskytuje kompletní informace o mapě server. Vzdáleně připojení boti však tyto informace díky Gamebots nemají, a proto nezbývá než to emulovat vlastními prostředky. Proto boti používají příkaz *EQUIPEMENT*, který slouží hlavně pro rychlejší zmapování nové mapy už během první hry. Po skončení činnosti bota se nasbírané informace o mapě přidají do mapového souboru. Při příští hře bot má tyto znalosti k dispozici už při prvním objevení se na mapě.

Sbírání předmětů. Boti se snaží primárně získat zbraň a potom brnění. Pokud bot spatří zbraň, je již ozbrojen, ale jeho spolubojovník není, ignoruje ji, aby mu dal možnost ji sebrat, na rozdíl od UT botů, kteří si je navzájem berou. Problém je s určením nejbližší zbraně. Momentální reprezentace mapy umožňuje určit absolutní vzdálenost vzdušnou čarou, což je ve většině map naprostý nesmysl (zbraň je za zdí v místnosti, do níž vede dlouhá cesta okolo a přitom je jiná zbraň hned ve vedlejší místností). Nebo můžeme určit počet navigačních bodů, které se nacházejí na cestě ke zbrani. Ovšem i zde narazíme na problém – omezená délka zprávy v GB. Při moc dlouhé cestě příkaz na vyžádání cesty k předmětu selže, takže se hodně vzdálené předměty tváří jako hodně blízké. Proto si bot volí náhodně ze známých předmětu s tím, že k nějaké zbraní určitě dojde, a když po cestě potká jinou, vezme ji.

5.6 POSH plán

POSH plán je patrně nejdůležitější částí implementace. Korektní zpracování jednotlivých smyslů a činností knihovny chování je samozřejmě pro fungování nezbytné, nicméně plán je sdružuje a uvádí v pohyb. Nejobtížnější je rozhodnout správnou posloupnost jednotlivých položek v kořenu rozhodovacího stromu. Položky se zde neustále po řadě kontrolují a pokud je některá splněná, vykoná se příslušná sekvence činností či se přejde na další úroveň rozhodování.

Velkou výhodou POSHe je, že lze velmi snadno modifikovat vytvořeného agenta a vytvářet různé mutace. V mém případě například vytvořím boty, kteří si předávají informace, ale neprovádějí taktické operace, jednoduchým odstraněním souvisejících podmínek z kořene plánu. Takže se dá testovat vliv libovolné akce či postupu na celkový projev bota, ale hlavně vytvořit odlišně se chovající boty modifikací plánu při zachování knihovny chování.

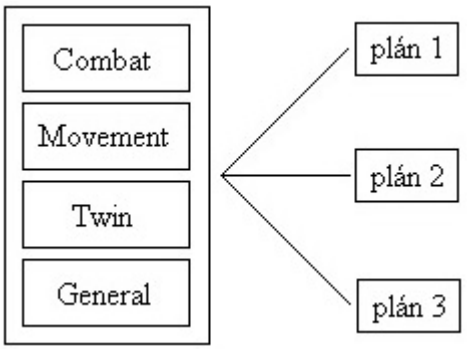


Fig. 12. Napojení několika různých plánů na stejnou knihovnu chování.

Kořen stromu pro spolupracující dvojici reprezentuje hlavní cíle a jejich priority. Při tvorbě POSH plánu jsem začal s následujícím náhledem na cíle a priority a postupným upravování a laděním jsem získal kořen ukázaný na Fig. 13.

Hlavní cíle a jejich priority (1 = nejvyšší)

1. náraz do stěny
2. nepřítel
3. zprávy a příkazy mezi spolubojovníky
4. zdraví bota
5. bytí spolu
6. sběr předmětů
7. pohyb po mapě

```
(RDC life (goal ((fail)))
 (drives
; GENERAL
 ((stucked (trigger ((is-stucked-long) (has-not-opponent) (not-see-enemy-long))) ap-caper (seconds 1)))
 ((wall (trigger ((hit-wall))) ap-jump (seconds 1)))
 ((unarmed (trigger ((impact-hammer) (know-weapon))) c-collect-weapon))
 ((stop-the-shooting (trigger ((is-shooting) (not-see-enemy-long) (has-opponent))) ap-stop-shooting-and-chase))
 ((stop-the-shooting2 (trigger ((is-shooting) (not-see-enemy-long))) ap-stop-shooting))
 ((get-armors (trigger ((see-armor) (not-see-enemy))) ap-get-armor))
; COMMAND PART
 ((cmd-received    (trigger ((not-see-enemy) (command-received))) ap-do-nothing)) ; just receives command
 ((send-opponent-info (trigger ((has-opponent) (see-enemy))) ap-send-enemy (seconds 2)))
; ENGAGE
 ((engage (trigger ((engage-started) (twins-together) (see-enemy))) ap-engage))
 ((stop-engaging (trigger ((engage-started) (twins-not-together))) ap-stop-engage))
 ((start-engage (trigger ((not-sent-engage) (twins-together) (see-enemy))) ap-send-engage))
; COMBAT
 ((respond-to-damage (trigger ((taken-damage) (has-not-opponent))) c-turn-to-damage (seconds 2)))
 ((respond-to-lost-opponent (trigger ((has-opponent) (not-see-enemy) (armed))) c-chase-opponent))
 ((respond-to-enemy (trigger ((armed) (see-enemy))) ap-respond-to-enemy))
 ((heard-noise (trigger ((not-see-enemy) (hear-alien-noise))) c-turn-to-noise (seconds 1)))
; GETTING NECESSARY ITEMS
 ((health-low (trigger ((see-med-kit) (low-health) (twin-health-ok))) ap-get-med-kit))
 ((get-weapons (trigger ((see-weapon) (not-see-enemy) (twin-armed))) ap-get-weapons))
; FOLLOW
 ((sending-follow (trigger ((havent-sent-follow) (twins-together) (i-am-the-one-and-only))) ap-send-follow))
 ((switch-follow (trigger ((is-following) (twins-together) (i-am-the-one-and-only))) ap-switch-follow (seconds
2)))
 ((do-following (trigger ((is-following) (twins-together))) ap-follow))
 ((not-together (trigger ((is-following) (twins-not-together))) ap-get-together))
; GO TO ITEMS
 ((go-to-some-med-kit (trigger ((very-low-health) (know-med-kit))) c-collect-med-kit))
 ((go-to-some-armor (trigger ((know-armor))) c-collect-armor))
; GET TOGETHER
 ((get-together (trigger ((twins-not-together) (havent-sent-follow))) ap-get-together))
; SOMETHING TO DO ANYTIME
 ((get-item (trigger ((see-item) (not-see-weapon) (not-see-med-kit))) ap-get-stuff))
 ((going-around-map (trigger ((enough-special-points))) ap-run-around-map))
 ((run-somewhere (trigger ((succeed))) ap-behej))
)))
```

Fig. 13. Hlavní část POSH plánu. Fungování je popsáno v textu.

Obrázek výše (Fig. 13) představuje kořen rozhodovacího stromu. Jsou v něm podle priority seřazena jednotlivá pravidla, která určují botovo chování. Pravidla jsou rozdělena, pro větší přehlednost, do několika bloků oddělených komentáři v kapitálkách. Rozhodování bota budu popisovat právě po těchto blocích.

První se zjišťuje (část *GENERAL*), zda bot dlouho nestojí, či zda nevrazil do zdi. Podle toho vyskočí, nebo se zkusí rozběhnout k nejbližšímu NP, nebo se otočí. Pokud není ozbrojen (má *ImpactHammer* – nejslabší zbraň ve hře), snaží se získat zbraň, tedy doběhnout k některé ze známých zbraní. Pokud není tísněn nepřítelem a vidí zbraň či brnění, vezme je. Pokud ztratil výhled na protivníka, přestane střílet a začne jej pronásledovat (běží na poslední známé místo výskytu).

Dále přijímá zprávy od spolubojovníka a pokud vidí nepřítele, posílá o něm informace (část *COMAND PART*).

Následuje pasáž pro taktickou operaci obestoupení – *ENGAGE*. Bot první zjistí, zda jsou splněny podmínky pro spuštění a pošle příslušný příkaz. Pak začne pohyb na stranu a pokouší se protivníka obestoupit.

Následující část je věnována boji – *COMBAT*. Pokud bot nevidí protivníka a někdo jej zraní, otočí se k místu, odkud zranění přišlo. Pokud jej vidí, následuje běžný útok na nepřítele. A na závěr pokud zaslechl zvuk, jehož původcem není jeho spolubojovník, ohlédne se.

Pokud ani jedno z předchozích pravidel neuspělo, bot není tísněn nepřítelem a tak se může věnovat dalším činnostem. Velmi důležité je udržovat botovi vysokou výdrž a dobré vybavení. Proto pokud zahlédne lékárničku, nebo zbraň, snaží se ji sebrat.

Další část je věnována společnému pohybu po mapě – *FOLLOW*. Když jsou boti spolu, zjistí si, který z nich je vhodnější pro vedoucí pozici (srovnají si výdrž a typ zbraně). Vhodnější bot pak pošle příkaz *FOLLOW* a druhý jej od té chvíle následuje. Pořadí může změnit druhý bot. Pokud zjistí, že je vhodnější on, pošle příkaz *ERASE FOLLOW*. Společný pohyb má poměrně nízkou prioritu, protože zajišťuje početní převahu botů při boji a možnost se navzájem krýt. Boj má vyšší prioritu a na společný pohyb v boji je část *ENGAGE*.

Část *GO TO ITEMS* se používá než se boti dostanou k sobě. Bot se snaží, pokud má nízkou výdrž, získat předměty zvyšující výdrž – lékárnička, brnění.

Bot, který došel s rozhodováním až k části *GET TOGETHER*, nebojuje, má dostatečnou výdrž, zbraň, takže už mu nic nebrání, aby se snažil dostat ke svému spolubojovníkovi a zahájit společný pohyb.

Na závěr jsou pravidla (*SOMETHING TO DO ANYTIME*), která zajišťují sběr předmětů, kolem nichž bot prochází, a kontinuální pohyb bota po mapě. Až na konec kořene plánu dojde jen vedoucí bot dvojice, druhý jej následuje, takže skončí v části *FOLLOW*.

Vývoj POSH plánu. Na představeném kořeni rozhodovacího stromu je vidět jistá odchylka od počátečního rozvrhnutí cílů a priorit. Plán vznikal postupným přibližováním se cílovému chování za průběžné implementace částí knihovny chování a ověřování korektnosti jednotlivých smyslů a činností. Zjistil jsem, že v některých místech je vhodnější jiné pořadí pravidel, než by se mohlo zdát na první

pohled. Místy se hrubě definované cíle prolínají.

Mutace POSH plánu. Jak jsem již zmínil, jednoduchými úpravami kořene plánu lze vytvořit jiným způsobem spoluporacující dvojice. Před zahájením testů jsem z plánu, jehož fragment je na Fig. 13, vytvořil další dva plány pro méně spolupracující dvojice.

První dvojice využívá pro spolupráci pouze výměnu informací a následné konání s ohledem na ně. Tedy jsem odstranil části, které souvisely se speciálními operacemi (*ENGAGE*, *FOLLOW*, *GET TOGETHER*).

Druhá dvojice botů navzájem nespolupracuje vůbec, takže bylo třeba odstranit pasáže, ve kterých se využívá komunikace a informace z ní získané. Vznikl plán pro autonomního bota, který nestřílí na své spojence.

Kapitola 6

Testy

6.1 Úvod

Cílem testů bylo zjistit vliv spolupráce na kvalitu botů. Pro tento účel jsem vytvořil další dva POSH plány a tedy tři rozdílně se chovající dvojice. Testy proběhly přímo v UT na třech vybraných mapách. Dílčími výsledky testů jsou výsledky jednotlivých odehraných her. Tedy hodnotí se převaha jedné dvojice nad druhou.

Při testování se ukázalo, že některé chyby platformy jsou opravdu závažné. Například občasné odpojování botů ze hry, nebo pád herního serveru. Tyto chyby nepřekáží při vývoji a odpojování agenta nebrání ani plynulé hře. Pro potřeby testů je však nutné, aby k těmto chybám nedocházelo a boti byli ve hře po celou dobu. Proto se testování výrazně protáhlo a testů je méně, než jsem původně zamýšlel.

Nejprve představím nastavení v němž jsem dvojice testoval, poté výsledky testů a tuto kapitolu uzavřu jejich interpretací.

6.2 Podmínky testů

Při testování jsem použil UT boty pro srovnání a tři různě spolupracující dvojice mých botů, které jsem porovnával. Jednotlivé kombinace jsem testoval v různých obtížnostech a na různých mapách abych zjistil i dopad těchto vlivů.

Boty jsem spouštěl při následujícím nastavení hry:

Gamespeed: 90%, Friendly Fire: 0 %, Frequency of bots: 12 Hz

Gamespeed určuje rychlost hry, ovlivňuje hlavně rychlost pohybu a střelby. *Friendly Fire* určuje část z celkového zranění, kterou udělí bot svému spoluhráči, pokud jej zasáhne (může jej i zabít a tak snížit skóre týmu – záporný bod). V první sérii testů jsem zvolil *Friendly Fire* 0% a v druhé naopak 100%, abych zjistil vliv tohoto parametru. *Frequency of bots* je parametr POSHe, který udává, na jaké frekvenci běží logika botů. Hodnota 12Hz je zvolena s ohledem na frekvenci příjmu zpráv od Gamebots (cca 10Hz).

Dále jsem vybral tři mapy. Byly vybrány dvě mapy vyhovující pohybovým schopnostem botů. Boti totiž neumí používat *movery* ani *Jump Pointy* a navíc nevnímají hrany propastí či lávy, takže tyto mapy jsou skoro jednoúrovňové a neobsahují problematické prvky. Třetí mapa je pravým opakem prvních dvou. Byla zařazena jen pro srovnání. Ukázala, že boti jsou použitelní na vhodných mapách a budou použitelní na všech typech, až bude jejich pohybový aparát kompletní. To však závisí na rozšíření GB.

Mapy byly před testy upraveny. Byly odstraněny problematické zbraně a ponechány pouze tři druhy. Pro demonstraci je to postačující, neboť dané zbraně jsou dostatečně různorodé – některé jsou vhodné na boj na blízku, jiné pro boj na větší vzdálenost. Zvoleny byly tyto zbraně:

– *minugun2* – rotační kulomet. Je vhodný pro boj na krátkou vzdálenost. Má poměrně velký rozptyl, takže se jeho střelám dá jen těžko uhnout. Navíc tak může zasáhnout několik u sebe stojících soupeřů zároveň.

– *ShockRifle* – energetická zbraň. Má dva typy střelby – kouli a paprsek. Při jejich kombinaci vyvíjí ničivou explozi.

– *SniperRifle* – odstřelovací puška. Hodí se zvláště na střelbu na větší vzdálenost.

Testy proběhly na těchto mapách:

– *Stalwart.* Jedná se o komplex místností. Je to jednoúrovňová mapa, ve které není nutné používat *movery* ani skákat. Je však poměrně členitá.

– *Agony.* Představuje jeskynní dvoupodlažní mapu. Podlaží jsou propojena dvěmi nakloněnými plošinami, takže není nutné používat *mover* (zde zdvižná plošina). Navíc jsem z mapy odstranil všechna brnění, abych zjistil i vliv tohoto faktoru na výsledky.

– *Morpheus.* Je to mapa, která je pro boty naprosto nevhodná. Skládá se ze tří věží, mezi kterými je třeba skákat, navíc věže mají několik pater. Na takovou mapu nedostačují pohybové schopnosti botů. Neumí skákat mezi věžemi.

Testu se tedy zúčastnily tyto typy botů:

1	UT Boti – originální boti z Unreal Tournamentu
2	boti, kteří spolu nikterak nekomunikují ani jinak nespolupracují
3	boti, kteří si předávají informace o předmětech a protivnících, ale neprovádějí taktické operace
4	boti, kteří provádějí i speciální týmový přístup k protivníkovi (operace *Engage*) a snaží se pohybovat společně.

Hra skončila pokud jeden z týmu dosáhl skóre čtyřiceti *fragů* (frag je bod získaný za zabití protivníka).

6.3 Výsledky

Výsledky 62 testů shrnují následující tabulky. V záhlaví jsou na řádcích jména map a ve sloupcích jsou vypsány typy zúčastněných botů (viz výše). Testoval jsem první na obtížnost *Skilled*, což je třetí nejnižší obtížnost, a *Friendly Fire* na 0%. Druhá sada testů byla na obtížnost *Masterfull*, což je třetí nejvyšší obtížnost, a *Friendly Fire* na 100%. Výsledky jsou zaneseny jako poměry týmových skóre.

Mapa / Boti	*1 vs. 2*	*1 vs. 3*	*1 vs. 4*	*2 vs. 3*	*2 vs. 4*
Stalwart	40:27	40:29	40:35	32:40	40:31
	40:19	40:25	39:40	40:28	40:36
	40:19	32:40	40:35	36:40	36:40
Agony	40:30	40:36	40:32	33:40	40:33
	40:34	34:40	39:40	32:40	35:40
	40:34	27:40	38:40	38:40	32:40

Tab. 1. Tabulka s výsledky testů na obtížnost *Skilled*.

Mapa / Boti	*1 vs. 2*	*1 vs. 3*	*1 vs. 4*	*2 vs. 3*	*2 vs. 4*
Stalwart	40:29	40:36	40:24	33:40	40:32
	40:39	40:17	40:14	32:40	40:31
	40:27	40:36	40:28	31:40	40:35
Agony	40:38	40:24	40:26	35:40	40:33
	40:31	40:28	40:34	40:36	35:40
	40:33	40:32	40:38	40:36	32:40
Morpheus	N/A	-1:-17	N/A	-10:-10	N/A

Tab. 2. Tabulka s výsledky testů na obtížnost *Masterfull*.

Boti	*1 vs. 2*	*1 vs. 3*	*1 vs. 4*	*2 vs. 3*	*2 vs. 4*
Součet	480:360	453:383	476:386	422:460	450:431
Rozdíl	120	70	90	- 38	19

Tab. 3. Srovnávací tabulka součtu všech výsledků a rozdílů mezi nimi.

6.4 Jak to vlastně dopadlo?

Dílčí výsledky samozřejmě nemůžeme chápat absolutně a je třeba je rozumně interpretovat (bohužel nešlo dost dobře aplikovat metody statistické analýzy, neboť nešlo spouštět testy automaticky a ve velkém množství). Výsledky jsou ovlivněny charakterem zvoleného módu hry. V módu Team Deathmatch se boti po smrti objevují na náhodném Spawn Pointu na mapě. Může se tedy velmi snadno stát, že se bot po smrti objeví skoro na stejném místě, kde před chvílí zemřel. Samozřejmě ho čeká patřičné uvítání a slabě vyzbrojený bot často podlehne početnějšímu, či lépe vybavenému protivníkovi. Navíc mapy nejsou příliš rozlehlé a tak dochází k častému kontaktu s nepřítelem. Proto může docházet k bodovým „šňůrám“, když se bot několikrát neobjeví zrovna na tom správném místě. Tedy lze říct, že remíza je, když hra skončila rozdílem menším než pět bodů.

UT boti. První se zaměřím na výsledky UT botů proti mým dvojicím (Tab. 1 a 2, první tři sloupce). UT boti jsou proti mým botům zvýhodněni. Dostávají informace rychleji a mají kompletní přehled o mapě (o pozicích předmětů). Vnímají projektily, které kolem nich létají, a dokáží se jim vyhnout. Také jsou schopni lepšího pohybu,

ale i oni nejsou bez chyb a občas zůstanou někde stát. Přes tyto faktory se s nimi mí boti dokázali poměrně solidně vypořádat.

Z pozorování v průběhu testů je patrné, že UT boti zvládají lépe sběr předmětů zvyšujících výdrž (brnění apod.). To má nezanedbatelný vliv na výsledné skóre. Na testech je to vidět na rozdílu mezi mapami Stalwart a Agony. Stalwart totiž na rozdíl od Agony obsahuje tyto předměty. UT boti byli úspěšnější právě na mapě Stalwart. Také zvýšení obtížnosti víc prospělo UT botům. Dokáží efektivněji využívat výhod z vyšší obtížnosti plynoucích (lepší rozhled, přesnost).

Při konfrontaci s UT boty vynikly také rozdíly mezi jednotlivými typy mých dvojic. Jednodušší boti se při zvýšení obtížnosti zlepšili, zatímco složitější zhoršili. Domnívám se, že je to způsobeno například tím, že UT boti častěji používají třeba kombinovanou explozi u Shock Rifle. Takticky spolupracující dvojice se drží víc pospolu, takže je vetší šance takto zabít oba najednou. Dalším důvodem pro zhoršení výsledků taktických botů na vyšší obtížnosti je to, že jsem na vyšší obtížnosti zapnul i *Friendly Fire*. Taktičtí boti se častěji střílí do zad, protože jsou častěji spolu a tedy si i častěji překážejí ve střelbě.

Největší roli na zlepšení výsledků všech typů dvojic proti UT botům měla implementace náhodných úhybů při boji, což alespoň trochu eliminovalo to, že boti nevnímají projektily.

Vyhodnocení různě spolupracujících dvojic. Velkým překvapením testů bylo, že si velmi dobře vedla dvojice nespolupracujících botů (Tab 3 – rozdíl pouhých 19 bodů proti takticky spolupracující dvojici). Z pozorování vysuzuji, že přečíslení není až tak velká výhoda a i osamocený bot může s poměrně slušnou pravděpodobností porazit dva boty. Záleží na konkrétní situaci, výzbroji, výdrži a také štěstí účastníků boje (boti nestřílí absolutně přesně). Díky tomu pro boty může být výhodnější samostatný pohyb, protože když se drží spolu, posbírají méně předmětů, potkají méně nepřátel a pokryjí menší část mapy. Společný pohyb (jak je vidět z pozorování) nedává takticky spolupracujícím botům jistotu úspěchu v situaci, kdy dojde na boj a budou stát dva proti jednomu protihráči.

Z toho by se dalo usuzovat, že spolupráce nemá smysl, ale to není pravda. Testy také ukázaly, že spolupráce na bázi informací nese ovoce. Zatímco rozdíly mezi takticky spolupracujícími a nespolupracujícímu boty jsou z hlediska výsledků her vyrovnané, boti spolupracující čistě pomocí výměny informací mají nad ostatními navrch.

Mapa Morpheus. Zde mí boti nebyli schopni rozumného pohybu a tedy testování nemělo jiný smysl, než ukázat, že boti nejsou schopni korektní práce na všech typech map. Zajímavá byla situace, která nastala při testování proti originálním UT botům. Po nějaké době se situace ve hře ustálila. Stabilizovalo se skóre (mí boti víc padali do propasti, a proto měli skóre -17) a boti utvořili dvě skupinky v odlehlých částech mapy, kde se pořád dokola pohybovali na malé ploše. Tedy ani UT boti nebyli schopni vhodně fungovat na této mapě a čekali na aktivitu soupeře.

Shrnutí. Nejefektivnější byli boti, kteří spolupracovali pouze na úrovni využití předávání informací (viz Tab. 3). Ukázalo se, že v daném nastavení nejsou zvolené

taktické operace přílišnou výhodou, ale nejsou přítěží. Pozorování v průběhu testů odhalilo, že početní převaha v boji nehraje v jednoduchém módu hry a za daných pravidel, výdrže a zbraní tak významnou roli.

Zajímavé by také bylo zařadit test uvěřitelnosti botů, tedy test proti lidským protihráčům. Pro tento účel se mi však nepodařilo získat dost dobrovolníků, kteří by se testování zúčastnili.

Kapitola 7

Příbuzné práce

Dnešní vývoj týmové souhry v FPS hrách se ubírá mnoha směry. Vědečtí pracovnící i vývojáři si uvědomují poptávku herní veřejnosti po vyspělejší týmové inteligenci. Provedení týmové inteligence a inteligence protivníků začíná významné ovlivňovat prodejnost jednotlivých herních titulů. S tím souvisí i vzrůst zájmu o tuto oblast a snaha vyvinout něco nového, lepšího.

Z příbuzných prací v oboru uvedu hlavně praktické implementace, na které jsem narazil při zkoumání tohoto tématu. Nenalezl jsem žádnou práci, která by se zabývala přímo módem Team Deathmatch, takže půjde o práce na stejných, či podobných platformách, ve kterých jsem hledal inspiraci a srovnání.

7.1 Týmová souhra

Z oblasti týmové souhry mně zaujaly tři projekty:

HIVEMind: Highly Interconnected Verbose Mind [14] je přístup aplikovaný na robotech pracujících v reálném prostředí. Spočívá v rozhodování o následné akci v závislosti na tom, co vnímá robot a jeho spolupracovníci. Robot tedy vnímá své spolupracovníky jako virtuální senzory, jejichž výstupy zohledňuje při svém rozhodování. Tím jsem se částečně inspiroval při tvorbě spolupracující dvojice botů.

Planning To Coordinate: Projekt [10] je založen na autonomních agentech jimž přiděluje úkoly externí plánovač SHOP na základě informací o stavu hry. Komunikace probíhá dle hvězdicové architektury, kdy plánovač je uprostřed. Boti spolu navzájem nekomunikují, jen provádějí příkazy plánovače a posílají mu informace o okolí. Tedy se jedná o aplikaci centralizovaného přístupu s tím, že "velitel" není účasten ve hře. Projekt byl implementován na spojení Gamebots a UT (mód hry *Domination*). Je zcela odlišný od mého projektu, nicméně vznikl na téměř stejné platformě a je zajímavý z hlediska potencionálních budoucích prací na softwarovém projektu CTF.

TEAM: The Team-oriented Evolutionary Adaptibility Mechanism [15] představuje použití centralizovaného hierarchického přístupu k řízení botů. Byl implementován na hře Quake III (hra typu FPS) mód hry CTF (podobné jako v UT). TEAM používá evoluční a adaptivní mechanismy. Vytvořená jednotka botů se proto dokáže přizpůsobovat hře protivníka. Po nějaké době tréninku už má nepřátelský tým (v tomto případě počítačový, který jedná často velmi podobně) hodně malou šanci na úspěch. Otázkou je, jak by se takový přístup zachoval při hře proti lidským protivníkům, nicméně je to zajímavý projekt a opět inspirace pro budoucí práci na CTF.

Volba přístupu závisí hlavně na problému, na nějž se jej snažíme aplikovat. Například jednoduchý přístup bez spolupráce je ještě dostačující pro hry typu TDM, ale je velmi nepřesvědčivý v případě módů Domination či CTF, kde je týmová hra klíčem k úspěchu. Stejně tak hierarchický přístup může snadno selhat v prostředí, které

se velmi často mění, neboť dlouhodobá rozhodnutí velitele skupiny mohou vést k chybným příkazům. Zajímavá je rovněž kombinace přístupů a zakomponování učení, rozpoznávání strategie soupeře a dalších technik.

7.2 Autonomní boti

Neméně důležitá, ne-li důležitější, je postava samotného bota. I zde můžeme nalézt řadu prací a aplikací, které se zabývají teoretickými tezemi i jejich uvedením do praxe. Uvedu zde dvě práce, které využily UT, Gamebots nebo POSH.

BOTbot: BOTbot představuje první pokus o komplexnějšího bota vytvořeného na spojení UT, Gamebots, pyPOSH. Vytvořil jej Samuel Partington [13] na universitě v Bath (UK). Jedná se však spíše o ukázku použití BOD a POSH než o obstojně fungujícího bota. Bot je sice schopen korektně hrát CTF v UT, ale není zde ani pokus o týmovou spolupráci a bot značně pokulhává i v běžných herních činnostech.

G O L O G bot. GOLOG bot [16] využívá původních UT botů. Umožňuje jejich ovládání pomocí speciálního jazyka ReadyLog. Autoři chtěli ověřit, zda je možno používat logic-based jazyky nejen na teoretické úrovni, ale i v zábavném průmyslu a toto se jim zde podařilo ukázat. Zajímavé bylo, že používali přímo originální boty z UT bez použití Gamebots, o čemž jsem před zahájením prací také uvažoval.

Existuje rovněž mnoho komerčních řešení této problematiky. Ty jsou však převážně veřejnosti nepřístupná, nebo řeší problémy v těsné součinnosti s jádrem hry. Lze se jimi jen inspirovat při vytváření vlastního autonomního chování.

Kapitola 8

Budoucí práce

Řešení týmové inteligence i obecných problémů v FPS hrách je pole dost široké a neorané. To mi otevírá mnoho možností pro další práci v této oblasti. Vyjma rozšiřování samotného projektu o další vlastnosti a vylepšování stávajících botů, připadá v úvahu několik dalších možností. Těm se budu podrobněji věnovat v následující kapitole.

8.1 Místnosti

Při návrhu taktických operací se doslova nabízí speciální provedení vstupu do místnosti, ústupu a jiných operací s využitím okolního prostředí k získání taktické výhody. Takové akce by pak mohly být vhodné u složitějších módů hry jako je CTF, Domination či Assault. Operace tohoto typu vyžadují taktické informace o mapě. Ty nejsou v základní reprezentaci, vytvořené z informací dodaných UT serverem, k dispozici. Velmi dobré by bylo například umět rozeznat vstup a výstup z místnosti. Je potencionálně několik možností, jak tento problém řešit, ovšem žádná není obecně vyhovující.

Prvním nápadem je třeba přidat do mapy u vstupů jiný typ navigačního bodu. Pokud dáme jen jeden, nezískáme tím nic, neboť z jednoho bodu nemůžeme zjistit polohu vůči „dveřím“. Tedy potřebujeme minimálně dva body. Ovšem bot má omezený úhel vidění a v některých situacích si nemusí ani jednoho bodu všimnout. Takže bychom mohli vždy počítat vzdálenost bota od jednotlivých přímek určenými speciálními body. Problém je s nekonvexními tvary místností.

Mimo jiné by i problém detekce místností měl vyřešit projekt NavMesh [6] mého kolegy Jakuba Gemrota, který je zaměřen na automatické získání taktických informací z geometrie mapy. Také proto jsem se zatím omezil na taktické operace, které nevyžadují dodatečné informace o okolí. Zařazení akcí založených na pokročilých informacích o mapě bude možné až po dokončení zmíněného projektu.

8.2 Capture The Flag[4]

Naše skupina sdružená okolo představeného spojení platforem uvažuje o práci na společném softwarovém projektu „*UT CTF Bots*“. Chceme vytvořit tým botů pro efektivní a zábavnou hru v UT v módu hry *Capture The Flag*. Při realizaci tohoto projektu chceme využít zkušeností získaných z práce na individuálních projektech – UT Emotion Bots [12] (boti s emocemi), UT Team Deathmatch Twins (týmová spolupráce) a UT NavMesh [6] (lepší reprezentace mapy). Naše projekty se totiž poměrně dobře doplňují a pokrývají podstatné problémy, kterými se budeme muset zabývat.

Boti by měli být díky emočnímu modelu méně předvídatelní a uvěřitelnější. Budou pracovat s pokročile reprezentovanou mapou, což jim umožní provádět různé taktické operace a rozumnější řešit problémy při pohybu.

4 Tato část byla převzata s úpravami ze článku [9]

Týmová inteligence bude řešena použitím centralizovaného přístupu. Bude existovat centrální arbitr, který povede globální strategii.

Problémy očekáváme zejména při tvorbě složitějších strategických operací a obecně při konstrukci řídícího arbitra. Dále bude nutné vyvážit emoční model tak, aby příliš často nenarušoval týmovou koordinaci.

8.3 Pogamut 2

Další variantou skupinového softwarového projektu by mohlo být rozšíření stávajícího frameworku Pogamut. Rozšíření by obnášelo přesun Gamebots na novější verzi Unreal Tournamentu (pravděpodobně UT 2004), vytvoření vlastních komunikačních kanálů a rozšíření ladícího prostředí Pogamut pro tvorbu botů hrajících CTF.

Stávající zkušenosti totiž ukázaly na vážné nedostatky popsaného spojení platforem, které je třeba před zahájením prací na rozsáhlejším projektu odstranit.

Kapitola 9

Závěr

Podařilo se mi navrhnout a implementovat decentralizovanou týmovou inteligenci pro módu hry Team Deathmatch na spojení platforem UT [1], GB [2], POSH [3], Pogamut [6]. Tento projekt otestoval možnosti těchto prostředí a jejich vhodnost pro vývoj umělé inteligence.

Zjistil jsem, že UT není tak vhodný, jak jsem se původně domníval. Při vývoji jsem neustále narážel na problémy typu nesystematicky pojmenované zbraně, nedosažitelné předměty, špatně navržené rozmístění NP na mapě apod. Pokud jsme si těchto problémů vědomi, dají se rozumně řešit. Hlavní překážkou je odpojování botů ze hry bez známých příčin. Pokud bych chtěl získat vypovídající údaje z testování, musel bych jím strávit nepřiměřeně mnoho času (řádově týdny - jeden korektní test trvá přibližně hodinu). Navíc při vývoji několika členného týmu pro CTF, kdy se testu účastní i 8 externích botů, je pravděpodobnost odpojení mnohem vyšší. Přitom testování je nezbytnou součástí vývoje a je důležité pro posouzení vhodnosti jednotlivých postupů.

Oproti tomu Gamebots jsou poměrně vhodní pro práci tohoto druhu. Pokud se omezíme na vhodné typy map, lze s jejich pomocí relativně snadno implementovat žádané chování.

POSH a BOD se ukázali v dobrém světle, neboť jsem byl schopen i bez předchozích zkušeností s vývojem umělé inteligence navrhnout a následně implementovat cílové chování botů. Přesto si musíme být vědomi některých slabin těchto architektur.

Zajímavé jsou výsledky testů. Asi nejpodstatnějším výsledkem je, že dvojice spolupracující jen na bázi výměny informací je pro zvolený mód hry nejlepším řešením. Tedy se ukázalo správné vsadit u módu Team Deathmatch (dále TDM) na decentralizovaný model spolupráce. Použití představených taktických operací už nemá v TDM tak pozitivní vliv. Je však pravděpodobné, že existují jiné taktické operace, které jsou v tomto módu použitelné. Dále se ukázalo, že má smysl klást důraz na kvalitní projev samotného bota. Dobré zpracování herních dovedností má významný vliv na jeho šance v boji.

Během práce na projektu jsem došel k názoru, že nejdůležitějším prvkem z hlediska hratelnosti je kvalita počítačových spoluhráčů. Hráč totiž přijde do kontaktu s nepřítelem vždy jen na velmi krátkou dobu. Boj trvá řádově vteřiny, takže si sotva stačí všimnout, že protivník provádí složitější taktické operace. Oproti tomu může dobře vnímat komunikaci ve veřejném komunikačním kanálu. Použití decentralizovaného přístupu k návrhu týmové inteligence umožňuje emulovat smysluplný rádiový provoz mezi protihráči. Stačí jej hráči zpřístupnit a může lépe vnímat jejich spolupráci.

Se spoluhráči, oproti protivníkům, tráví hráč většinu času, a proto by mohlo mít smysl zakomponovat do hry i spolupráci na bázi informací mezi hráčem a jeho počítačovými spolubojovníky.

Literatura

[1] Epic MegaGames: *Unreal Tournament home page*, http://www.unrealtournament.com, [15. 5. 2006].

[2] Gamebots: *home page*, www.planetunreal.com/gamebots, [15. 5. 2006].

[3] Joanna J. Bryson: *Inteligence by design: Principles of Modularity and Coor dination for Engineering Complex Adaptive Agent*. PhD Thesis, MIT, Departement of EECS, Cambridge, MA, June 2001.

[4] Joanna J. Bryson: The behavior-oriented design of modular agent intelligence. In: *Agent Technologies, Infrastructure, Tools, and Applications for e-Services* (R. Kowalszyk, Jörg P. Müller, H. Tianfield, and R. Unland, eds.), Springer, 2003.

[5] Joanna J. Bryson: *homepage*, www.cs.bath.ac.uk/~jjb, [15. 5. 2006].

[6] Jakub Gemrot: *Pogamut & NavMesh homepage*, http://ail.jinak.cz/, [15. 5. 2006].

[7] pyPOSH: *homepage*, www.cs.bath.ac.uk/~jjb/web/pyposh.html, [15. 5. 2006].

[8] William van der Sterren: *Squad Tactics: Team AI and Emergent Maneuvers*, In: AI Game Programming Wisdom (Steve Rabin, ed.), 233-246, Charles River Media, 2002.

[9] M. Bída, O. Burkert, C. Brom, J. Gemrot: *Pogamut – Platforma pro prototypování botů v Unreal Tournamentu,* In: Sborník příspěvků Kognice a Umělý život VI, Česká Republika, Třešť, 2006.

[10] Hai Hoang: *Planning To Coordinate: using HTN to Coordinate Unreal Tournament Bots,* 2005.

[11] Andy Kwong,: *A Framework for Reactive Inteligence through Agile Component-Based Behaviors*. Master's thesis, Department of Computer Science, University of Bath, 2003.

[12] Michal Bída: *homepage*, http://aiproject.wz.cz, [15. 5. 2006].

[13] Samuel J. Partington a Joanna J. Bryson: *The Behavior Oriented Design of an Unreal Tournament Character*, Proceedings of IVA 2005, Springer, Berlin, 2005.

[14] Aaron Khoo and Ian Douglas Horswill: *HIVEMind : Grounding Interference in Cooperative Activity*, AAAI 2001 Fall Symposium on Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems, 2001.

[15] Sander Bakkes, Pieter Spronck, Eric Postma: *TEAM: The Team-oriented Evolutionary Adaptability Mechanism*, ICEC, Eindhoven, 2004.

[16] Stefan Jacobs, Alexander Ferrein and Gerhard Lakemeyer: *Unreal G O L O G Bots*, 2005.

Appendix 1 – Obsah CD

Na přiloženém CD se nalézají:

- uživatelská, instalační a vývojová dokumentace projektu
- videa – ukázky činnosti botů
- upravené mapy, na nichž se odehrálo testování
- článek [9]
- aktuální verze Gamebots
- Pogamut
- zdrojový kód projektu
- tato práce ve formátu pdf

Doporučuji zejména shlédnout videa a k nim přiložený soubor s komentáři k zachycenému dění ve hře.