Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Michal Bída

Emoční boti v prostředí hry Unreal Tournament

Katedra software a výuky informatiky

Vedoucí bakalářské práce: Mgr. Cyril Brom

Studijní program: Obecná informatika

2006

Rád bych na tomto místě poděkoval Cyrilu Bromovi za vedení. Dále bych rád poděkoval svým rodičům, kteří nejednou cennou radou přispěli k napsání této práce a Jakubu Gemrotovi, který byl hlavním tvůrcem frameworku Pogamut.

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

V Praze dne 9. 8. 2006 Michal Bída

Obsah

Název práce: Emoční boti v prostředí hry Unreal Tournament
Autor: Michal Bída
Katedra: Katedra software a výuky informatiky
Vedoucí bakalářské práce: Mgr. Cyril Brom
e-mail vedoucího: brom@ksvi.mff.cuni.cz

Abstrakt: Práce se zabývá využitím emocí v umělé inteligenci v počítačových hrách. Bude zkoumán možný přínos emocí pro umělou inteligenci z hlediska lepší imitace lidského chování. Cílem práce je implementace emočního modelu v prostředí hry Unreal Tournament (projekt UT Emotion Bots) a zhodnocení jeho vlastností a vhodnosti pro simulaci emocí v FPS hrách. Práce dále představí platformy použité při vývoji UT Emotion Bots a zhodnotí jejich vhodnost pro vývoj umělé inteligence.

Klíčová slova: umělá inteligence, boti, emoce, FPS hry

Title: Emotion bots in the enviroment of the game Unreal Tournament
Author: Michal Bída
Department: Department of Software and Computer Science Education
Supervisor: Mgr. Cyril Brom
Supervisor's e-mail address: brom@ksvi.mff.cuni.cz

Abstract: This work is concerned with usage of emotions in artificial inteligence in computer games. It inspects possible benefits of emotions for artificial inteligence in the means of better imitation of human behavior. Main goal of this work is the implementation of an emotion model in the enviroment of the game Unreal Tournament (project UT Emotion Bots) and appraisal of its properties and suitability for the simulation of emotions in FPS games. This work introduces platforms used in the development of the project UT Emotion Bots and it evaluates their suitability for development of artificial inteligence.

Keywords: artificial inteligence, bots, emotions, FPS games

1 Úvod

V poslední době stoupá počet lidí, kteří se zabývají vývojem umělé inteligence v počítačových hrách. Tento trend je patrný zejména v komerční oblasti, ale dle mého názoru se dá pozorovat již i na akademické půdě (rok 2006). Na Matematicko-fyzikální fakultě Univerzity Karlovy v Praze se jedná o zvýšený počet seminářů, které se zabývají umělou inteligencí, a které jsou zatím nepravidelně organizovány některými vyučujícími (například Seminář z umělých bytostí organizovaný Mgr. Cyrilem Bromem).

Umělá inteligence v počítačových hrách slouží k řízení počítačem kontrolovaných entit. Tyto entity spadají do dvou hlavních kategorií. Jedná se o entity, které jsou reprezentované postavou fyzicky přítomnou v počítačové hře, a entity, které postavou ve hře reprezentované nejsou (může se jednat například o algoritmy, které řídí skupinu jednotek ve strategiích; jejich hlavním cílem je především umožnit hráči hrát hru bez lidských protihráčů).

Jedním z cílů umělé inteligence je také zprostředkovat hráči lepší herní zážitek. Ve hrách, kde se vyskytuje hodně počítačem řízených postav, se toho dá dosáhnout vytvářením iluze, že všechny postavy ve hře jsou skutečnými lidmi (mají vlastní osobnost, cíle apod.). V mé práci budu v souvislosti s tímto používat termín uvěřitelnost, kterým budu označovat míru toho, jak dobře daná postava imituje chování (ať už lidské nebo nějaké fiktivní bytosti).

Tato práce se bude zabývat využitím emocí v umělé inteligenci v počítačových hrách typu FPS (FPS – First Person Shooter). Jedná se o akční hry v 3D prostředí, které hráč vidí z vlastního pohledu postavy, za kterou hraje. Práce předkládá projekt UT Emotion Bots, ve kterém se pokusím využít emoce jako prostředek ke zlepšení uvěřitelnosti počítačových protivníků v FPS hře Unreal Tournament [1] (dále UT). V projektu UT Emotion Bots bude použit emoční model vycházející z modelu Alexe J. Champandarda, který je prezentován v knize [2].

Projekt využívá platformu, která se skládá z FPS hry Unreal Tournament, reaktivně plánovací architektury POSH [3], rozhraní GAMEBOTs [4] a frameworku Pogamut [5]. Projekt je spjat s projektem Ondřeje Burkerta UT Team Deathmatch Twins [6] (decentralizovaná spolupráce počítačových protivníků ve hře Unreal Tournament) a projektem Jakuba Gemrota UT Navmesh [7] (lepší reprezentace herní lokace ve hře Unreal Tournament). Naše projekty a platformy byly představeny na konferenci KuZ v Třešti v článku [8].

1.1 Motivace

Emoce ovlivňují lidské chování v mnoha aspektech. Umožňují člověku se rychle a efektivně rozhodovat, slouží jako filtr informací, které člověk přijímá svými smysly, podstatně ovlivňují fungování lidské paměti a zároveň mají mnoho dalších

efektů [9]. Přesto všechno byly emoce donedávna ve spojitosti s umělou inteligencí opomíjené. Nyní už je situace jiná a vznikají první emoční modely. Tyto modely často vycházejí z poznatků učiněných v rámci psychologie, která se lidskou osobností a emocemi zabývá.

Přes vzestupný trend v oblasti využití emocí v umělé inteligenci nejsou ještě emoční modely ve hrách typu FPS příliš využívané. Dle Alexe J. Champandarda [2] je umělá inteligence v těchto hrách často podceňována a důraz je kladen především na jednoduchost a efektivnost. Proč by počítačoví protivníci v FPS hrách měli být schopni imitovat lidské emoce?

Několik let jsem se pohyboval v komunitě hráčů hrajících FPS hry a na základě nejen mých zkušeností můžu konstatovat, že emoce mají podstatný vliv na hráčovo chování a jeho celkový výkon ve hře. Ilustruji to nyní na několika příkladech.

Nervozita a sebedůvěra. Nervozita a sebedůvěra ovlivňují zejména přesnost akcí jednotlivých hráčů (přesností zde myslím, nakolik dokonale jsou akce prováděny). Může se jednat o schopnost celkově spolupracovat s ostatními hráči, ale jsou ovlivněny také jednodušší akce, jako například zaměřování cíle (při střelbě). Se stoupající nervozitou se přesnost akcí snižuje, hráči také začínají častěji chybovat (zvolí špatné místo pro úkryt apod.). Sebedůvěra má naopak přesně opačný efekt.

Hněv a strach (respekt). V přírodě ovlivňují tyto dvě emoce hlavně rozhodnutí mezi útěkem a bojem. U hráčů v FPS hrách je to podobné. Rozzlobený hráč se často rozhodne pro boj i v situacích, kdy je to pro něj nevýhodné (nepřítel je lépe kryt, má početní převahu apod.). Často to působí až komicky, když se takový hráč bezhlavě vrhne se slabou zbraní do klubka nepřátel a je okamžitě zabit. Naopak hráč, který cítí ke svým soupeřům respekt, volí častěji variantu útěku (doprovázenou například ještě krycí palbou). Otevřený boj volí takový hráč většinou pouze tehdy, má-li nad svým soupeřem nějakou výhodu (silnější zbraň, soupeř ho nevidí apod.). Často se stává, že se takovýto hráč zdržuje na výhodných pozicích, kde své soupeře očekává.

Emocí, které zásadněji ovlivňují chování hráčů v FPS hrách, je více. Mohl bych jmenovat ještě například radost a únavu (skleslost). Pro pochopení toho, jak moc emoce hráče ve hrách ovlivňují, jsou dle mého názoru zmíněné projevy dostačující.

Domnívám se, že imitací emočních jevů vyskytujících se v FPS hrách u člověka můžeme dosáhnout větší uvěřitelnosti chování u počítačových protivníků. Dále se domnívám, že je výhodné simulovat tyto emoční jevy pomocí emočního modelu vycházejícího z poznatků z psychologie, neboť dojde k přiblížení emočních reakcí počítačového protivníka k lidským. V současné době existuje více různých emočních modelů s různým zaměřením. Ve své práci jsem se rozhodl vycházet z emočního modelu Alexe J. Champandarda a to zejména proto, že je zaměřen na implementaci v FPS hrách.

1.2 Cíle práce

Hlavní cíle práce jsou:

1. Implementace emočního modelu vycházejícího z modelu Alexe J. Champandarda v projektu UT Emotion Bots
2. Zhodnocení implementace emočního modelu z hlediska vhodnosti pro simulaci emocí v počítačovém protivníkovi a návrhy zlepšení.
3. Zhodnocení celkového zlepšení uvěřitelnosti počítačového protivníka a popsání faktorů, které uvěřitelnost ovlivňují (možnosti platforem, emoční projevy atd.).

Jako vedlejší cíl jsem stanovil zhodnocení všech platforem použitých při implementaci projektu UT Emotion Bots z hlediska jejich vhodnosti pro vývoj umělé inteligence v počítačových hrách (jedná se o prezentování zkušeností, které jsem získal při práci na projektu).

2 Uvěřitelnost

V této kapitole rozeberu trochu více termín uvěřitelnost (tak jak jsem jej definoval), neboť se jedná o klíčový pojem této práce.

Uvěřitelnost protivníka v FPS hře je ovlivněna mnoha faktory. Podle toho, které faktory bereme v úvahu, můžeme uvěřitelnost rozdělit například na uvěřitelnost rozhodování, uvěřitelnost pohybu nebo na uvěřitelnost verbálního projevu. Některé faktory nemůže programátor ovlivnit (omezení dané platformami apod.). V mé práci jsem se soustředil především na uvěřitelnost emočního projevu a na uvěřitelnost rozhodování, které na základě emočního modelu bot činí (výraz bot se používá pro označení počítačového protivníka v FPS hře). V souvislosti s tímto jsem zjistil, že emoční model může snižovat předvídatelnost botových akcí, přičemž může zachovat uvěřitelnost takového chování.

Vysoká míra předvídatelnosti akcí botů v FPS hrách je poměrně obvyklý jev. Snížit předvídatelnost botů a přitom zachovat konzistentnost a uvěřitelnost chování by mohlo přispět k větší oblibě takových her. Věřím, že emoční model jako řešení tohoto problému může dosáhnout dobrých výsledků a v závěru práce věnuji těmto úvahám pár řádků.

3 Projekt UT Emotion Bots

Projekt emočních botů si klade za cíl implementovat emoční model v počítačovém protivníkovy v prostředí FPS hry Unreal Tournament (Fig. 1). Spíše než na výkon bota v počítačové hře byl v tomto projektu kladen důraz na implementaci, popis a zhodnocení funkce implementovaného emočního modelu.

Při práci na projektu jsem vycházel z template třídy bota vytvořené v rámci Pogamutu (více dále). Template třída z Pogamutu představuje základní kostru bota, která se stará o zpracovávání zpráv z prostředí a vykonávání příkazů. Na tuto třídu můžeme nahlížet jako na vnější reprezentaci herní lokace v UT (pomocí proměnných a symbolických struktur). Součástí této třídy nejsou žádné řídící struktury. Bot, který by byl vytvořen jen pomocí této třídy, by bez dalšího kódu jen přijímal zprávy z prostředí a nijak na ně nereagoval (nehýbal by se, nestřílel by, neprováděl by žádné akce). Hlavním cílem template třídy Pogamutu je ulehčit práci programátorovi, který se nemusí starat o komunikaci s UT.

Kromě implementování emočního modelu bylo nutné implementovat veškeré botovo chování. Toto chování mělo za cíl řešit problémy pohybu po mapě, sběru předmětů a boje s protivníky. Botovo chování bude podrobněji popsáno v kapitole 6.4 Emoční chování.

Fig. 1. Prostředí hry Unreal Tournament. Přejato z [8]**.**

Z technického hlediska využívá projekt platformy Pogamut [5]. K prostředí hry Unreal Tournament je připojena reaktivně plánovací architektura POSH („Parallel-rooted, Ordered Slip-stack Hierarchy“) [3], v níž je psána řídící struktura bota (zahrnující všechna chování). Součástí projektu je rozhraní GAMEBOTs, které slouží jako prostředník v komunikaci mezi UT a ostatními použitými platformami. Vlastní emoční model je pak psán v jazyce python [10]. V další kapitole budou popsány jednotlivé platformy zmíněné výše.

4 Platformy

V této kapitole popíšu všechny platformy použité při práci na projektu UT Emotion Bots. Jedná se o FPS hru Unreal Tournament, rozhraní GAMEBOTs, architekturu POSH a framework Pogamut.

4.1 Unreal Tournament

Unreal Tournament je FPS hra odehrávající se v 3D simulaci reálného světa. Obsahuje desítky herních lokací, tzv. *map* – různých simulovaných míst, kde se hra odehrává. Příklady map jsou zříceniny, podzemní pevnosti, vesmírné základny, ale také třeba jedoucí vlak.

Z hlediska projektu UT Emotion Bots jsou důležité následující pojmy:

- **Avatar (postava ve hře)** – jedna postava nacházející se v herní lokaci. Může být ovládána lidským hráčem nebo se může jednat o počítačového protivníka – bota. Každý avatar má několik základních vlastností, které se mohou v průběhu hry měnit. Jedná se o:
  - *Výdrž* („health“), která udává, kolik poškození je ještě daná postava schopná snést než zemře.
  - *Brnění* („armor“), které ovlivňuje míru poškození, které postavě způsobují zbraně ve hře.
  - *Zbraň* („weapon“), kterou postava právě drží (jedna postava může vlastnit více zbraní, držet v jednu chvíli však může jen jednu z nich).
  - *Rychlost pohybu* („speed“), kterou se avatar právě pohybuje.
- **Navigační body („Navigation Points“, „Navpoints“ nebo „Way Points“)** – Při pohybu po herní lokaci využívají boti tzv. navigační body, které jsou pro lidské hráče normálně neviditelné. Tyto body reprezentují místa, kam se lze s avatarem dostat a můžeme na ně nazírat jako na jakousi síť, která propojuje různá místa na mapě. Tato síť bodů, rozmístěných na herní lokaci, slouží botům jako hrubá reprezentace dané lokace.
- **Deathmatch, Team Deathmatch** – Unreal Tournament obsahuje více různých módů hry. Každý mód nějakým způsobem modifikuje pravidla a cíle hry. Boti z projektu UT Emotion Bots jsou určeni pro hru v módu *Deathmatch* nebo *Team Deathmach*. Boti fungují i v ostatních módech hry, nejsou ale schopni plnit některé speciální požadavky, které na ně ostatní módy kladou. Dále popíšu jen módy *Deathmatch* a *Team Deatchmatch*.
  - V *Deathmatchi* hrají všichni proti všem (boti i hráči). Úkolem je eliminovat co nejvíce soupeřů. Pokud je hráč zabit soupeřem, objeví se znovu na jiném místě herní lokace se základní konfigurací zbraní, výdrže a brnění.
  - *Team Deathmatch* se od klasického *Deathmatche* liší tím, že hráči hrají proti sobě v týmech. Cílem se pak stává eliminovat co nejvíce soupeřů v rámci

týmu.

- **UnrealEngine, UnrealScript, UnrealEd –** Počítačová hra Unreal Tournament je postavena na enginu UnrealEngine [11], což je virtuální stroj inspirovaný Java Virtual Machine. UnrealEngine je zaměřen na 3D vizualizaci objektů a prostředí. Programy pro tento stroj se píší v UnrealScriptu. UnrealScript je jazyk, který byl vytvořen firmou Epic. Samotný jazyk je inspirován Javou a C++. Součástí hry Unreal Tournament je také UnrealEd, což je nástroj pro tvorbu nových lokací a je možné v něm editovat i všechen kód hry napsaný v UnrealScriptu (tedy i například originální boty dodávané s hrou).

4.1.1 Originální boti v Unreal Tournamentu.

Hra Unreal Tournament obsahuje boty naprogramované v jazyce hry UnrealScript. Jejich logika (řídící struktury, které je ovládají) je založena na hierarchických konečných automatech[1]. Boti reagují na dění ve hře prostřednictvím zpráv, které jim zasílá herní server. Zprávy jsou zasílány prostřednictvím volání metod ve třídě bota. To znamená, že při příchodu zprávy se spustí určitá metoda (funkce, procedura apod.), která je s danou zprávou asociována. Tyto metody mohou být v různých stavech bota implementovány různě, tedy se jeho reakce může lišit podle toho, v jakém stavu se nachází. Například řekněme, že je bot ve stavu flee (únik) a je na něm zavolána metoda „vidím nepřítele“ (SeeEnemy). Protože je bot ve stavu flee, změní směr svého pohybu směrem od nepřítele. Kdyby ale byl například ve stavu attack (útok), zachoval by se pravděpodobně jinak (změnil by směr pohybu směrem k nepříteli).

Dle mého názoru bylo hlavním cílem při tvorbě originálních botů naprogramovat efektivní počítačové protivníky s dobře nastavitelnou obtížností. Boti v Unreal Tournamentu dokážou vykonávat poměrně dost rozličných, komplikovaných činností, které zahrnují například krycí palbu při ústupu, doplnění střeliva v probíhajícím boji nebo pronásledování protivníka směrem nejpravděpodobnějšího úniku. Boti jsou schopni hrát jakýkoli z originálních módů hry a plnit rozličné úkoly, které jim tyto módy ukládají. Velmi rychle dokážou reagovat na zprávy z prostředí a umí spolu v omezené formě spolupracovat.

V čem originální boti zaostávají? Jedná se například o předvídatelnost chování. Ve stejných situacích zareagují vždy stejně. Tohoto se dá velmi dobře využít při boji proti nim. Boti nedisponují žádným emočním modelem (nemají žádné emoce). Využívají stále stejné cesty na mapě, po kterých se pohybují vždy stejným způsobem (přeskočí překážku vždy na stejném místě apod.). Některé cesty a přístupy do místností jsou pro ně tímto uzavřeny. Boti neumí hráče obejít nebo mu vpadnout do zad. Pokud se zdá, že to udělají, jedná se o emergentní jev[2].

1 Hierarchické konečné automaty – Jedná se o strukturu, která je reprezentována určitou skupinou obecných stavů. Jednotlivé stavy se od sebe liší, aktivní může být v jednu chvíli jen jeden stav. Struktura je hierarchická, protože jeden stav se může dále rozkládat na více podstavů.

2 Emergentní jev – Jedná se o jev, který není v botovi explicitně implementován.

4.2 GAMEBOTs

Kapitola byla převzata z [8].

GAMEBOTs (GB) [4] je klient-server aplikace napsána v UnrealScriptu, která slouží k ovládání avatara v Unreal Tournamentu. GB je server, ke kterému je možné se připojit pomocí TCP/IP protokolu. GB poskytuje rozhraní pro vytvoření nového bota v Unreal Tournamentu a jeho další ovládání. GB definuje textový protokol, pomocí něhož lze botovi dávat příkazy a ze kterého se dovídáme informace o okolí bota. GB zasílá botovi pouze informace o tom, co právě vidí nebo co může slyšet. Bot není vševědoucí, např. neví, jestli se za rohem někdo schovává nebo ne.

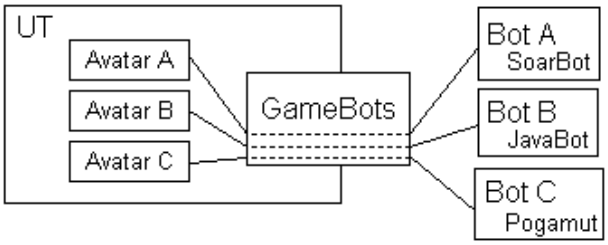

Fig. 2. GB spojuje avatara s různými logikami, které jsou psány v různých jazycích. Převzato z [8]

GB zprávy

Klienti si s GB vyměňují jednotlivé zprávy. Máme tyto tři typy zpráv:

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

Velikou výhodou tohoto rozhraní je zvolený model klient-server, který umožňuje napsání logiky bota v jakémkoli jazyce (Fig. 2) a který obsahuje podporu síťové komunikace. Také umožňuje spouštět logiku bota na jiném počítači než server Unreal Tournamentu.

UT Emotion bots bude fungovat na jakékoli platformě, která bude pro komunikaci s externími nástroji využívat rozhraní GAMEBOTs. UT Emotion Bots je tedy přenositelný v rámci herních platforem podporujících rozhraní GAMEBOTs. Toto zahrnuje zejména řadu her Unreal Tournament, Unreal Tournament 2003 a Unreal Tournament 2004, které využívají jazyka UnrealScript.

4.3 POSH

POSH („Parallel-rooted, Ordered Slip-stack Hierarchy“) je architektura pro tvorbu řídících algoritmů autonomních agentů. Vyvinula ji J. Bryson na University of Bath [12]. POSH architektura byla implementován A. Kwongem [13] v pythonu a zkušebně propojena s UT (PyPOSH).

POSH je symbolická, reaktivní architektura, která staví na paradigmatu „behavior-oriented-design“ (dále BOD), designu orientovaného na chování, který vychází z Brooksova přístupu k řízení robotů [14], nicméně Bryson ho pro potřeby své architektury rozšiřuje.

BOD vidí agenty jako uskupení semi-autonomních modulů nazývaných „behaviors“. Každý z těchto modulů má vlastní paměť a vlastní skupinu smyslů (senzorů) a aktů (efektorů). V POSHi nad nimi pracuje centrální interpret, který jim na základě tzv. *POSH plánů* (Fig. 3) předává řízení. Jednotlivé moduly ale mohou pracovat i paralelně.

```
(
 (AP otoc-se-a-pozdrav (seconds 1) (otoc-se-k-hraci zamavej-rukou rekni-ahoj ) )

 (C toulej-se-lokaci (seconds 1) (goal((vzdy-nesplneno)) )
  (elements
   (
    (uhybej-stenam ( trigger (( naraz-do-steny ))) poodejdi-od-steny )
    (drz-se-navigacnich-bodu ( trigger (( vidis-navigacni-bod ))) jdi-k-nav-bodu)
   )
  )
 )

 (RDC life (goal ((vzdy-nesplneno)))
  (drives
   ((zdrav-hrace ( trigger ((vidis-hrace) (hrac-pritel)) ) otoc-se-a-pozdrav (seconds 5)))
   ((prozkoumavej-lokaci ( trigger ((vzdy-splneno)) ) toulej-se-lokaci))
  )
 )
)
```

Fig.3. Příklad POSH plánu. Obsahuje kompetenci (C) sloužící k dekompozici chování, hlavní strukturu plánu (RDC life) a jeden action patern (AP) používaný k zápisu sekvenčních akcí.

Z technického hlediska lze POSH plán vidět jako rozhodovací pravidla uspořádaná do stromové struktury. Pravidla se zapisují v jazyce připomínajícím lisp. V této práci bude použita pythonovská implementace POSHe, ve které jsou jednotlivé behaviorální moduly implementovány jako objekty v jazyce python. Smysly a akty modulů se implementují jako metody v těchto objektech.

4.4 Pogamut

Pogamut je framework propojující dříve zmíněné platformy UT, GAMEBOTs a POSH (Fig.4.). Je naprogramován v jazyce python a pro svůj grafický interface využívá knihoven Javy, které jsou zpřístupněny díky jythonu. Kromě grafického interface obsahuje třídu UTBot, která je vlastně venkovní reprezentací bota z UT. Třída obsahuje a pravidelně aktualizuje všechny dostupné informace o světě v UT zasílané prostřednictvím GAMEBOTs a její součástí je i rozsáhlá skupina metod sloužících pro ovládání bota v UT. Programátor se tedy vůbec nemusí starat o komunikaci s UT a omezuje se pouze na volání metod třídy UTBot. GUI Pogamutu vycházelo z PyPOSH GUI Andyho Kwonga [13].

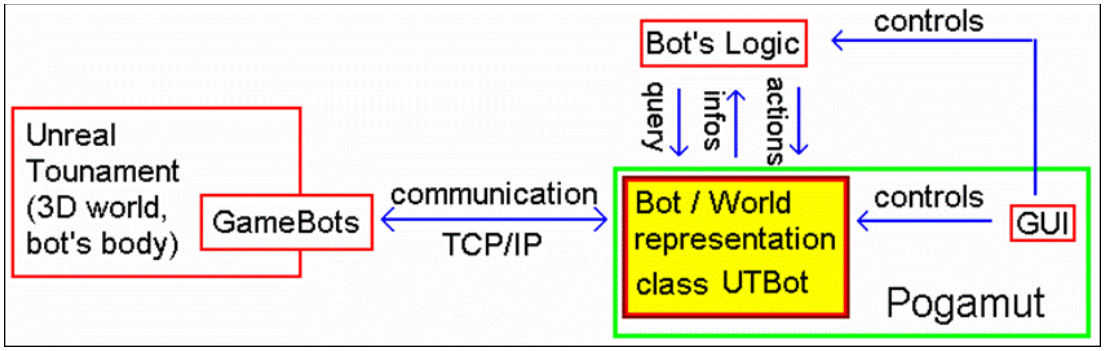

Fig. 4. Propojení platforem pomocí frameworku Pogamut. Obrázek převzat z [15]

Velkou výhodou frameworku Pogamut je, že umožňuje využít libovolný nástroj pro tvorbu řídících struktur bota tak, že zajistí jeho propojení s ostatními používanými platformami. Toto propojení je realizováno pomocí pythonovských tříd. V mém projektu používám propojení s POSHem.

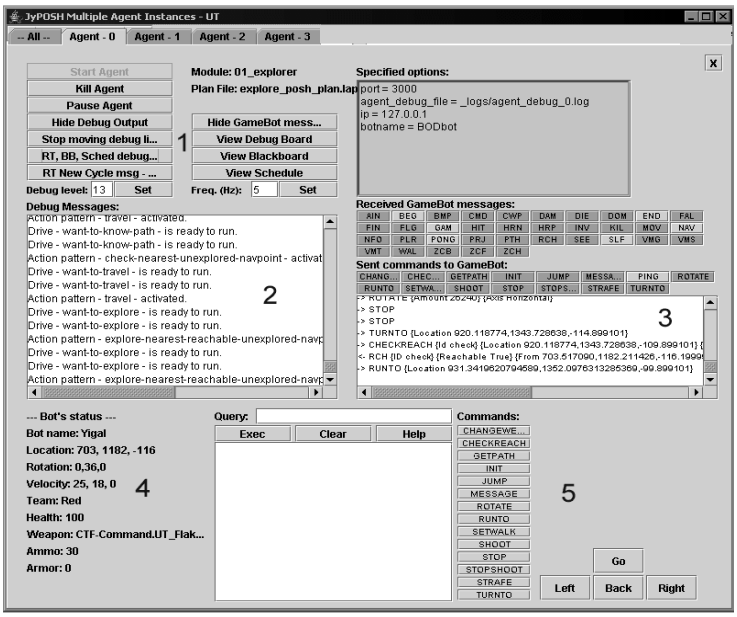

Fig. 5. Ukázka GUI pro ovládání instancí botů. Obrázek byl převzat z [8]

Ke každému botu existuje záložka (Fig. 5), která nabízí tyto možnosti:

- 1 – ovládání instance bota
- 2 – žurnál zpráv logiky bota
- 3 – žurnál komunikace mezi třídou UTBot a GB
- 4 – základní informace o botovi (jméno v UT, výdrž, zbroj, pozici, atd.)
- 5 – ruční ovládání bota (posílání příkazu GB)

5 Emoční model

V této kapitole popíšu model Alexe J. Champandarda, který mi sloužil jako vzor, ze kterého jsem vycházel při implementaci emočního modelu v projektu UT Emotion Bots. V další části této kapitoly popíšu emoční model použitý v projektu UT Emotin Bots.

5.1 Champandárdův model (vzor)

Champandardův model prezentovaný v [2] vychází z Plutchikova kruhového modelu emocí [16] (Fig.5.). Podle Plutchika existuje osm primárních emocí, které tvoří vzájemně se vylučující páry. Je to očekávání a překvapení, radost a smutek, náklonnost a odpor, strach a hněv. Podle jeho teorie je nemožné, aby člověk cítil dvě vzájemně se vylučující (komplementární) emoce ve stejnou chvíli. Plutchikovi emoce mohou mít různou intenzitu a mohou se vzájemně kombinovat do složitějších

celků. Například radost a náklonnost je chápána jako láska.

Champandardův model s drobnými úpravami převzal primární emoce z Plutchikova kruhového modelu a mimo emocí rozeznává ještě několik dalších základních entit, které budou vysvětleny dále. Jedná se především o pocity, city a nálady. Důležité jsou také interakce mezi těmito entitami. Následuje Chamandardův pohled na emoce.

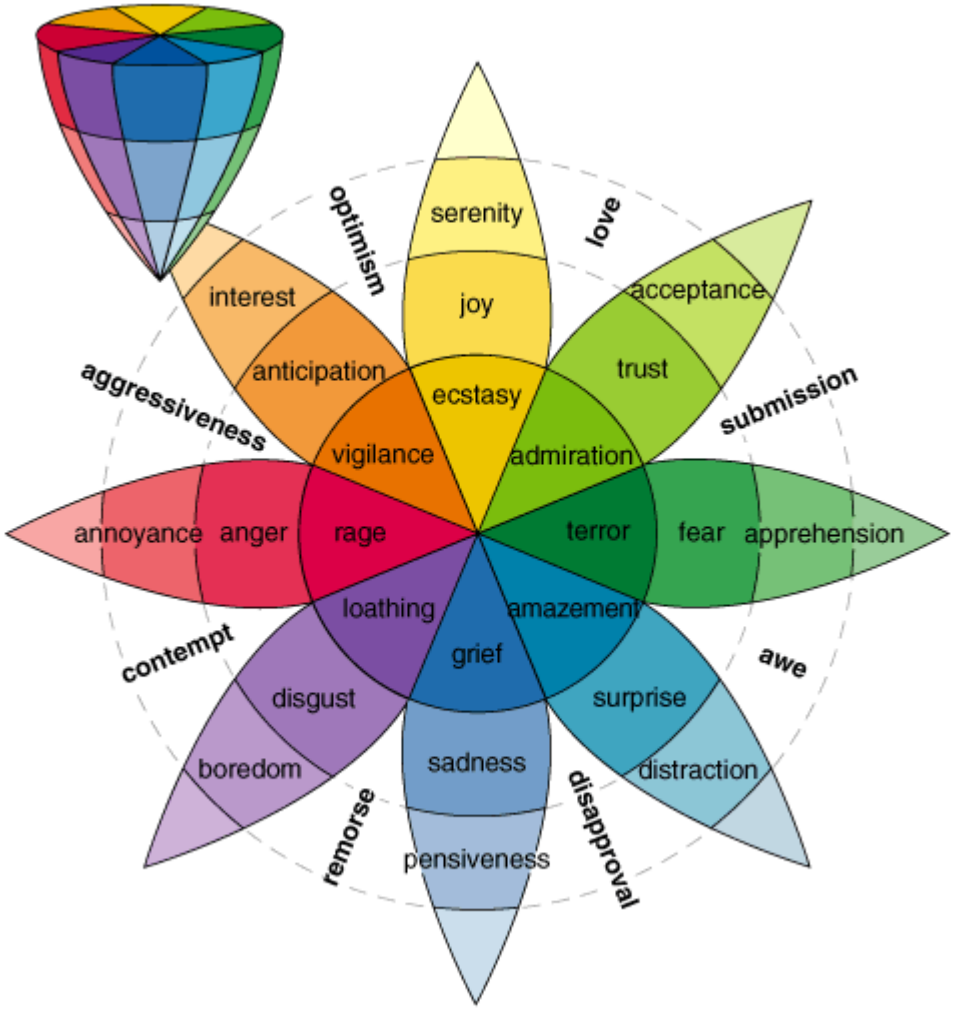

Fig. 5. Plutchikův kruhový model emocí.

Emoce vznikají interakcí těla a mozku a jsou ovlivňovány stimuly z okolí. Mozek reaguje na tyto nízko-úrovňové stimuly a vytváří z nich komplexní, trvalý stav mysli. Takovéto *emoce* se potom mohou projevit i navenek, např. strach útěkem.

Všechny *emoce* existující v nějaké bytosti jsou vyvolány *pocity*. *Pocity* můžeme chápat jako okamžitou reakci na momentální stav bytosti. *Pocity* jsou zažívány okamžitě jako reakce na změnu momentální situace a nemusí být nutně vyvolány reakcí na okolí, ale také vnitřně (myšlením).

Obvykle jsou *pocity* vyvolány vjemy. Tělo obdrží určitý stimul z okolí a

mozek jej interpretuje pomocí *pocitu*. Například NPC (Non Player Character – postava v počítačové hře ovládaná počítačem) může cítit překvapení, když se hráč objeví náhle, nečekaně. *Pocity* mohou být také vyvolány jako reakce na myšlenkový stav dané bytosti (například znalostí světa). V takovém případě vyvolává pocit obyčejné zpracování informací (například nějaký objekt není na svém místě, tak jak bylo očekáváno). Co mají *pocity* vyvolané stimuly z okolí a myšlením společné? V obou případech jsou vyvolány na základě přiřazení určitých vzorů v mozku.

Nálada je kompletní množina emocí, která reprezentuje momentální stav určité bytosti. Obvykle je nálada reprezentována jako množina několika primárních emocí, každá s jiným aktuálním stavem. Takováto kombinace může vyprodukovat složitější emoci (například lásku, optimismus).

Cit je trvalá asociace nějaké *emoce* s nějakým objektem. Je to mnohem silnější koncept – na rozdíl od *pocitů* nezávisí úplně na momentálním stavu. *City* jsou místo toho spjaty s objekty v minulosti a v budoucnosti. *City* jsou velmi užitečné, můžeme jimi vyjádřit například nenávist k určité osobě, zbrani apod.

Implementace Champandardova modelu

Pro samotnou implementaci emočního rozhraní Champandard doporučuje použít konečné automaty. Podle něj jsou, tak jak jsou definovány, úplně ideální strukturou pro modelování emocí v prostředí počítače. Formální definice konečného automatu (dále KA) je:

$$KA=\{Q,L,q_0,delta,F\}$$

Je to tedy pětice skládající se z Q - což je množina všech stavů konečného automatu, L - což je množina symbolů, které mohou stát na vstupu, q_0 - počáteční stav, delta - množina přechodových funkcí mezi jednotlivými stavy na základě vstupních symbolů a F - množina konečných stavů. Nebudeme se nyní příliš zabývat přesným popisem toho, co může být vstupem KA, nebo jakého tvaru mohou být jednotlivé stavy ani tím, jak se KA reprezentuje v počítači. Místo toho se podívejme, jak se dá KA využít k modelování emočního chování v umělé inteligenci.

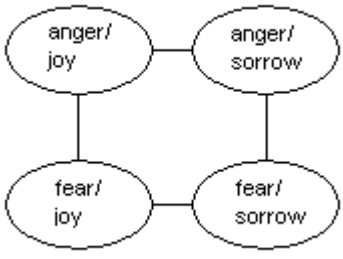

Fig.6. Příklad jednoduchého konečného automatu reprezentujícího 4 emoce (anger – hněv, fear – strach, joy – radost a sorrow – smutek).

Emoce mohou být konečným stavem. Řekněme, že chceme reprezentovat čtyři emoce pomocí KA. Automat bude mít čtyři stavy a vymodelujeme ho do čtverce (Fig.6.). Každá emoce je spjata se dvěma stavy a každý stav obsahuje dvě

emoce. Protichůdné emoce potom stojí na opačných stranách tohoto čtverce, čímž se zajistí, že nemohou nastat simultánně. Všechny stavy jsou propojeny se všemi ostatními. Podmínky k přechodům mezi stavy jsou vytvořeny tak, aby byly obecně použitelné a jsou definovány pomocí pocitů. Návrh tohoto KA je tedy zmrazen. Veškerá flexibilita bude vycházet z KA modelujícího pocity, který bude měnit aktivní emoce. Pro dobré fungování celého systému je důležitá integrace. Z venkovního prostředí přicházejí informace zpracovávané KA modelujícím pocity a zprávy o změně pocitů jsou potom posílány dál KA, který modeluje emoce. S tímto přístupem je emoční stav vždy aktuální, řízen reaktivně velmi efektivní cestou. Aby mohl tento model ovlivňovat chování bota, musí být nějakým způsobem propojen s modelem, který toto zajišťuje.

5.2 Emoční model v UT Emotion Bots

Emoční model v UT Emotion Bots přebírá z Champandardova modelu emoce, pocity, city a zároveň přechodové rovnice, na základě kterých se z pocitů generují emoce. Dále bude následovat seznam všech emocí, pocitů, citů a přechodových rovnic, které jsou použity v emočním modelu v UT Emotion Bots. V práci budu používat české názvy jednotlivých emocí, pocitů a citů.

Emoce použité v emočním modelu v UT Emotion Bots (na jednom řádku jsou dvě vzájemně komplementární emoce):

Radost (joy)	**Smutek (sorrow)**
Hněv (anger)	**Strach (fear)**
Pýcha (pride)	**Stud (shame)**
Pobavenost (amusement)	**Utahanost (weariness)**

Pocity použité v emočním modelu v UT Emotion Bots:

Překvapení (surprise)
Očekávání (anticipation)
Přitažlivost (attraction)
Odpor (disgust)
Bolest (pain)
Rozkoš (pleasure)
Zmatenost (confusion)
Objevení (discovery)

City použité v emočním modelu v UT Emotion Bots:

Přitažlivost (attraction)
Odpor (disgust)
Nenávist (hatred)
Lítost (pity)

Přechodové rovnice v UT Emotion Bots:

Rozkoš + očekávání = stimulace pýchy
Bolest + zmatenost = stimulace studu
Zmatenost + odpor = stimulace strachu
Bolest + překvapení = stimulace hněvu
Odpor + zmatenost = stimulace smutku
Rozkoš + přitažlivost = stimulace radosti
Objevení + překvapení = stimulace pobavenosti
Očekávání + zmatenost = stimulace utahanosti

Popis funkce emočního modelu v UT Emotion Bots:

Emoční vyhodnocování zpráv z prostředí probíhá pomocí pocitů, které tak zprostředkovávají přímou reakci na změny v prostředí. Toto vyhodnocování probíhá postupně a dá se říci, že simuluje práci konečného automatu. Pocity nemohou vznikat samovolně, generují se jen při přijmutí nějaké zprávy. Na základě pocitů se generuje případná změna emocí. Vždy dva aktivované pocity způsobují možné zvýšení (stimulaci) jedné emoce (viz. tabulka). Velikost stimulace může být v modelu ovlivněna (může být tedy ovlivněna pravděpodobnost, s jakou stimulovaná emoce přejde do vyššího stavu). Po stimulaci emoce jsou příslušné dva pocity deaktivovány, to znamená nastaveny na hodnotu 0.

Emoce jsou reprezentovány pomocí konečného automatu. Jeden automat reprezentuje jeden pár komplementárních emocí. Emoce slouží k reprezentaci emočního stavu bota, podle nich se upravuje botovo chování. V modelu může být aktivních více emocí naráz, nikdy ale nemůžou být aktivní obě emoce z komplementárního páru najednou.

City ovlivňují generování pocitů při přijímaní zpráv z prostředí. Jsou využívány jako dodatečná informace a můžou zásadně měnit množinu vygenerovaných pocitů. City mohou být asociovány pouze s ostatními hráči ve hře, nemohou být asociovány s předměty. City vznikají na základě herních statistik[3], které si bot během hry průběžně ukládá. Některé atributy statistik se v čase mění – například počet úmrtí protivníka, jiné zůstávají po celou dobu stejné – jméno protivníka, přitažlivost (míra toho, jak se botovi daný protivník líbí; tento atribut je

3 Herní statistika – soubor atributů, které jsou průběžně o hře ukládány. Jedná se například o počty úmrtí a zabití jednotlivých hráčů ve hře, míru úspěšnosti střelby a podobně.

náhodně vygenerován při prvním setkání s daným protivníkem), atd.

Parametrizace modelu zahrnuje výběr zpráv a událostí, na které bude bot reagovat. Dále je nutné specifikovat, jaké pocity s jakou hodnotou budou tyto zprávy generovat. Musí být také učiněno rozhodnutí, které atributy z herních statistik budou použity pro generování citů a specifikovat, jak se budou tyto city na základě takto získaných údajů vytvářet. V modelu je možné ovlivnit míru stimulace emocí. Všechna zmíněná rozhodnutí nejsou jednoduchá a nelze je přesně specifikovat, neboť se řešení těchto problémů může lišit případ od případu. Při parametrizaci modelu je nutné brát zřetel na cíl, který chceme pomocí implementace modelu dosáhnout.

Příklad práce emočního modelu

Práci emočního modelu nyní ilustruji na příkladu. Řekněme, že bot byl zasažen střelou a je zraněn, přičemž ale v okamžiku zásahu neviděl žádného protivníka. Z prostředí UT přijde zpráva o zranění, na kterou se zareaguje vygenerováním pocitu bolest. Jelikož bot nikoho neviděl, kromě pocitu bolest se vygeneruje také pocit překvapení. Kombinace pocitů bolesti a překvapení může s určitou pravděpodobností způsobit zvýšení excitace emoce hněv.

Z dlouhodobého hlediska funguje model takto: Čím častěji je hněv stimulován, tím pravděpodobněji může přejít do vyššího excitovaného stavu. Na druhou stranu, pokud by byla stimulována komplementární emoce ke hněvu – „strach“ – došlo by ke snížení pravděpodobnosti přechodu do vyššího stádia emoce hněv a ke zvýšení pravděpodobnosti přechodu do vyššího stavu emoce strach. Nemůže tedy dojít k tomu, že by byly obě emoce ve vyšším stádiu. Pokud jedna do vyššího stavu přejde, pak stimulace komplementární emoce způsobí nejprve snížení první emoce do základního stavu a až následně zvýšení excitace komplementární emoce.

Projevy emočního stavu bota budou zahrnovat změnu chování, změnu jeho některých vlastností, změnu verbálního projevu bota (textový kanál ve hře UT) a gestikulaci. Problematika projevů emocí v FPS hrách, stejně jako použité emoční projevy v projektu UT Emotion Bots budou podrobněji rozebrány v následující kapitole.

6 Projevy emocí

S emočním modelem úzce souvisí zvolené projevy emocí. Je zbytečné implementovat složitý emoční model, pokud se emoce nebudou dostatečně projevovat. V této kapitole nejprve představím Champandardův pohled na problematiku zobrazování emocí v FPS hrách, poté zhodnotím možnosti zobrazování emocí v použitých platformách a na závěr přejdu k výčtu použitých projevů emocí v projektu UT Emotin Bots.

6.1 Champandardův pohled na projevy emocí

Lidé vyjadřují emoce několika hlavními způsoby.

- **Výraz v obličeji –** nese neuvěřitelné množství informací o momentálním stavu člověka. Některá rozdělení primárních emocí jsou založena právě na rozpoznávání výrazů v obličeji.
- **Gesta –** řeč těla a gesta jsou také velmi silnou indikací emocí. Například přikyvování jako znak souhlasu.
- **Chování –** projevující se v delším časovém úseku. Například ignorace někoho jako známka odmítnutí.
- **Jazyk** – způsob jakým si lidé vybírají slova je velmi silnou indikací jejich momentálního duševního rozpoložení. Také celková stavba věty je ovlivněna emocemi. Krátké věty jsou více autoritativní a mohou být známkou hněvu.
- **Hlas** – tón a barva hlasu také reflektují momentální náladu. Zvyšování hlasu bývá indikací hněvu apod.

V počítačových hrách většinou nemáme k dispozici všechny tyto prostředky k zobrazování emocí. Většinou je množina možností, jak zobrazit emoce, silně omezená. Například tón a barva hlasu téměř vždy odpadají. Budeme se snažit používat takové způsoby prezentace emocí, které mohou být dobře vnímány ostatními hráči. V případě FPS hry, může mnoho akcí zobrazovat emoce. Vnímání světa (smysly) může být emocemi také ovlivňováno.

Základní dovednosti jako pohyb jsou ideální pro zobrazování emocí. Např. pomalá chůze může být známka deprese, velmi rychlý běh indikuje strach. Otočení směrem k objektu může znamenat náklonnost, směrem od něj odpor. Skákání může reprezentovat radost. Jiné akce nemusí být tak vhodné pro zobrazování emocí. Například změna zbraně, sbírání předmětů apod. nechává jen malý prostor pro zobrazení emocí. Na druhou stranu vyšší principy přítomné při rozhodování mohou zobrazit emoce. (například výběr slabé zbraně, aby byl protivník zesměšněn.) Nedostatek činnosti může být také zobrazením emoce (melancholie). Rozhodování na vyšší úrovni může také zobrazovat emoce – například výběr boje před útěkem.

Smysly hrají také velmi podstatnou roli v zobrazování emocí. Ovlivňují tok informací směrem od prostředí do rozhodovacích struktur. Emoce ovlivňují hlavně vnímání. Smysly mohou být méně ostražité v určitých kontextech emocí (například nezájem). V těchto případech nemusí bytosti vnímat ani blízké objekty tak dobře. Na druhé straně, pokud je bytost překvapena nebo polekána, její smysly budou velmi citlivé i k tomu nejmenšímu pohybu.

Manifestací emocí tedy mohou být například: Radost – skákání, tanec, mávání. Strach – menší přesnost při otáčení, ale zase zvýšená rychlost simulující zvýšenou hladinu adrenalinu. Panika – zastavení se na místě. Nuda – zmenšení přesnosti veškerých aktivit. Nechuť – otočení se zády. Zvýšení reakční doby při překvapení a naopak snížení při očekávání. Útěk před silnějším nepřítelem. A mnoho dalších.

6.2 Možnosti zobrazování emocí podle platforem

V této kapitole krátce zhodnotím možnosti použitých platforem z hlediska zobrazování emocí.

6.2.1 Emoce v Unreal Tournamentu

Ve hře Unreal Tournament nebyl na projevování emocí kladen důraz. Na druhou stranu Unreal Tournament je díky UnrealScriptu velmi dobře rozšiřitelný a už ve své základní verzi obsahuje spoustu věcí navíc, které nejsou normálně ve hře využívány. Jedná se například o podporu velké množiny nativních gest. Gesta v Unreal Tournamentu jsou určité pevně dané animace, které může postava ve hře provádět, zavolá-li se určitá funkce s příslušnými parametry. Jedná se například o mávání rukou, různé druhy skoků (bot se nehýbe z místa, to musí být zajištěno jinak), různé druhy uhýbání a také například o gesta typu vítězný tanec (animace zobrazující radost z úspěchu).

Kromě gest mají boti a hráči možnost posílat textové zprávy, které se pak zobrazují v herním okně. Změny stylu komunikace mohou indikovat velmi dobře změny emocí.

6.2.2 Emoce v POSHi

POSH je v projektu využíván pro tvorbu řídících struktur bota (logiky bota). Řeší tedy ovládání bota na vyšší úrovni. Možnosti vyjadřování emocí na této vyšší úrovni zahrnují zejména pozměnění priorit v botově chování, změnu způsobu vykonávání určitých rutinních akcí (například pohyb po mapě) a přidání dalších emočně specifických akcí (například číhání na protivníka při emoci stud). Projevy emocí implementované pomocí architektury POSH mohou být ve hře dobře patrné, záleží ovšem na tom, do jaké hloubky se budou rozdíly mezi jednotlivými emocemi implementovat.

Hlavní problém POSHe co se týče vyjadřování emocí je v tom, že POSH má fixní priority svých pravidel. Priority se za běhu programu nemůžou dynamicky měnit. Je nutné na začátku rozhodnout, která emoce bude mít přednost před ostatními. Tento problém se dá částečně řešit složitějšími smysly. Pomocí poznatků z psychologie můžeme navíc dávat přednost takovým emocím, které mají v přírodě obvykle větší váhu (například hněv přebije utahanost).

6.2.3 Emoce v GAMEBOTs

Aplikace GAMEBOTs pouze zprostředkovává komunikaci mezi Unreal Tournamentem a externími nástroji používanými při vývoji bota. Sama o sobě tedy nepřináší žádné další možnosti pro zobrazování a uchovávání emocí. Může pouze podporovat množinu zpráv a příkazů, které umožní využití všech dostupných způsobů pro zobrazování emocí ve hře Unreal Tournament.

6.2.4 Emoce v Pogamutu

Pogamut je framework propojující ostatní platformy a zefektivňující sdílení dat mezi jednotlivými platformami. Sám neobsahuje žádnou přímou podporu emocí. Nicméně součástí frameworku je třída generující statistiky ze hry (například počet zabití a úmrtí našeho bota, frekvence potkávání s ostatními boty apod.). Statistiky ze hry se dají využít při generování citů k ostatním hráčům ve hře. Vrstva citů je pro emoční model velmi důležitá, bez ní by nemohly být simulovány vztahy k ostatním objektům ve hře.

6.3 Projevy emocí použité v UT Emotion Bots

V projektu UT Emotion Bots jsem rozdělil všechny použité projevy emocí do čtyř skupin. Jsou to komentování dění ve hře, gestikulace, změna chování a upravení vlastností bota.

- **Komentování dění ve hře.** Bot je schopen na základě událostí ve hře komentovat dění ve hře. Komentování probíhá pomocí textového kanálu, který je přítomný v Unreal Tournamentu a do kterého lze posílat textové zprávy. Bot má připravený soubor textových zpráv. Každá zpráva slouží ke komentování nějaké události při určité excitované emoci. Často je pro určitou situaci a emoci k dispozici více možných zpráv, bot potom vybírá z těchto zpráv náhodně. Součástí komentování je i kontrola toho, aby bot nekomentoval dění příliš často a nedošlo k zahlcení komunikačního kanálu. K tomu botovi slouží soubor časovačů.(více v části 7 Implementace)
- **Gestikulace.** Do rozhraní GAMEBOTs jsem doprogramoval podporu některých nativních gest. Jedná se o gesta mávání (wave) a vítězný tanec (victory dance). Tyto gesta jsou vykonávána při splnění určitých podmínek ve hře. Jejich volání zajišťují smysly a akty architektury POSH. Gesto mávání se volá v případě, že bot potkal dosud neznámého hráče a má excitovanou emoci radost. Vítězný tanec je prováděn v případě, že bot úspěšně zlikviduje svého protivníka a má excitovanou emoci pýcha nebo radost.
- **Změna chování.** Pozměnění priorit v botově chování implementované pomocí přejití na jiný plán v architektuře POSH. Každá emoce má svůj vlastní více či méně modifikovaný plán. Neutrální chování a jednotlivá emoční chování budou představena dále v části 6.4 Emoční chování.
- **Upravení vlastností bota.** Byl vytvořen příkaz pro GAMEBOTs, kterým je možné ovládat rychlost pohybu bota. Rychlost bota bude upravována na základě emocí pobavenost a utahanost. Při excitované emoci pobavenost bude bot utíkat, při excitované emoci utahanost bude jen pomalu chodit. V případě, že není excitována ani jedna z těchto emocí bude se bot pohybovat naposled nastavenou rychlostí.

Důvodem, proč jsem vybral právě tyto čtyři projevy emocí, je to, že jsou ve hře dobře viditelné. Kromě toho jsou všechny tyto projevy více či méně podporované UT a nebylo tedy nutné dělat nějaké zásadnější zásahy do kódu hry. Změna chování

byla zvolena kvůli jednoduché implementaci pomocí architektury POSH.

6.4 Emoční chování

V této kapitole popíšu chování, která bot ve hře používá. Základem je neutrální chování, které je aktivní, pokud není excitována žádna emoce. Při popisu emočních chování (chování asociovaných s určitou excitovanou emocí) se omezím pouze na popis hlavních rozdílů proti neutrálnímu chování.

Kvůli architektuře POSH bylo nutné přesně určit prioritu jednotlivých emočních chování. Nejvyšší prioritu mají chování asociované s emocemi hněv a strach, následuje pýcha a stud a nakonec radost a smutek. K emocím pobavenost a utahanost není asociováno žádné speciální chování. Tyto emoce však ovlivňují rychlost pohybu bota ve hře.

Neutrální chování

Při vytváření bota jsem vzal v úvahu základní situace, které bude muset řešit. Vytvořil jsem jednu sadu chování pro každou z těchto situací. Základní situace zahrnují pohyb po mapě, když bot nevidí žádného protivníka, boj v případě, že protivníka vidí a útěk, v případě, že má bot málo života, nebo nějaký jiný důvod (např. taktickou nevýhodu apod.). Tyto tři základní situace jsou reprezentovány třemi kompetencemi (hlavní struktury POSH architektury sloužící pro zápis chování) v POSH plánu.

Kompetence wander-around. Je spuštěna v případě, že bot nevidí žádného protivníka. Toto chování zahrnuje náhodný průzkum mapy, sbírání předmětů a reagování na zvuky, střelbu či nárazy do stěny. Chování se snaží zamezit tomu, aby se bot zasekl u zdi nebo u nějaké překážky. Navigace po mapě probíhá pomocí navigačních bodů. Cílem tohoto chování je průzkum mapy a vybavení bota zbraněmi a brněním.

Kompetence fight. V případě přítomnosti protivníka se spouští kompetence fight, která reprezentuje bojové chování jako střelbu, uhýbání a pronásledování protivníka. Součástí tohoto chování je určení ideální vzdálenosti pro střelbu a udržování této vzdálenosti za současného uhýbání střelbě protivníka. Hlavním cílem tohoto chování je zlikvidovat protivníka a zamezit zlikvidování bota.

Kompetence flee. Chování v kompetenci flee se zase snaží dovést bota do bezpečí útěkem pro lékárničku nebo prostě směrem od nebezpečí. Toto chování se aktivuje při nízké hladině botova zdraví. Kromě úniku do bezpečí může zahrnovat například i krycí palbu.

Emoce pobavenost a utahanost

Tyto dvě emoce jsou jako doprovodné k ostatním. Nemají žádné speciální chování, které by s nimi bylo asociováno. Ovlivňují jen komentování dění ve hře (v případě, že nejsou aktivní žádné jiné emoce) a mění rychlost pohybu bota. Pokud je aktivní pobavenost, pak bot běží, pokud je aktivní utahanost, tak se bot pohybuje

pomalu. Bohužel Unreal Tournament nepodporuje zadávání rychlosti při úkrocích, takže při vykonávání úkroku bot vždy běží. Pravidla pro změnu rychlosti bota mají v POSH plánu nejvyšší prioritu.

Emoce hněv

Tato emoce výrazně zjednodušuje botovo chování. Útočí na všechny nepřátele, nevšímá si kvality držené zbraně, střeliva ani hodnoty zdraví. Při boji se vždy snaží dostat co nejblíže (ignoruje optimální vzdálenosti u zbraní). Když nevidí žádného nepřítele, tak ho z předmětů zajímají pouze zbraně. Rozzlobený bot nikdy neutíká.

Emoce strach

Bot, který má excitovanou emoci strach, nebojuje za žádných okolností. Bot pouze utíká směrem k lékárničkám od všech protivníků. Pohyb po mapě, pokud nevidí nepřítele, probíhá chaoticky a má reprezentovat celkovou vyděšenost bota. Často se rozhlíží a mění směr pohybu. Předměty sbírá normálně, sleduje tím zvyšování svých šancí na přežití.

Emoce pýcha

Pokud má bot excitovanou tuto emoci, pak se mu ve hře zatím dařilo. Bot se snaží dobře vybavit, když nevidí žádného protivníka. Pokud mu schází předměty jako zbraně nebo brnění a zná jejich polohu, tak se snaží přemístit na lokaci, kde se předměty nacházejí. V boji si bot důvěřuje, což se projevuje tak, že utíká až když má opravdu málo zdraví. Využívá optimálních vzdáleností pro útok se zbraněmi. Pokud někoho zlikviduje, spustí se gesto victory dance (vítězný tanec).

Emoce stud

Vyskytuje se zejména, pokud bot často prohrává souboje. Bot se opět snaží dobře vybavit. Získává zbraně a brnění stejně jako při excitované emoci pýcha. Rozdíl je v tom, že bot považuje za silné, jen opravdu ty nejsilnější zbraně a straní se konfliktů, pokud takovou zbraň nemá. Utíká už při známkách prvních zranění. Obsahuje speciální druh chování tactical-camp. Když je toto chování aktivní, bot číhá na určitém místě na své soupeře.

Emoce radost

Emoce radost nijak neovlivňuje boj a útěk bota. Ovlivňuje pouze pohyb po mapě, když bot nevidí protivníka. Bot skáče na předměty, které chce sebrat. Občas tančí, jako kdyby s někým bojoval. Když vidí nového hráče, tak ho pozdraví a spustí gesto wave – zamává mu. Pokud někoho zabije, tak spustí gesto victory dance – vítězný tanec.

Emoce smutek

Pokud má bot excitovanou emoci smutek, funguje vcelku normálně při boji nebo úniku. Ovlivněn je pohyb po mapě, když bot nevidí žádného protivníka. Bot si vůbec nevšímá předmětů (kromě lékárniček) a občas se zastavuje. Chování by mělo reprezentovat celkovou sklíčenost bota.

7 Implementace

Základní třídou celého projektu je třída *knight_bot.* Tato třída dědí třídu *UTBot* z frameworku Pogamut (Fig.7.). Třída *knight_bot* obsahuje všechny informace z prostředí zasílané prostřednictvím GAMEBOTs, slouží jako vnitřní reprezentace světa Unreal Tournamentu. Uvnitř třídy se nachází také další pomocné struktury a proměnné nutné pro správné chování bota.

```
class UTBot
    class knight_bot (
        class SensationsClass,
        class SpeechClass,
        class TimerClass,
        class CombatInfoClass,
        class PositionInfoClass,
        class emoce
    )
```

Fig. 7. Třídní návrh projektu UT Emotion Bot.

V třídě *knight_bot* probíhá konstrukce všech ostatních používaných tříd s výjimkou behaviorálních modulů (nutných pro architekturu POSH), které jsou konstruovány jinde. Třída *knight_bot* konstruuje třídy: *SensationsClass*, *SpeechClass*, *TimerClass*, *CombatInfoClass* a *PositionInfoClass* a využívá pomocnou třídu *utilities*. Třída *knight_bot* obsahuje též 4 třídy uchovávající emoce (viz. emoční model). Dále bude následovat stručný popis jmenovaných tříd.

SensationsClass slouží pro emoční zpracovávání informací z prostředí. Generuje změny emocí a city. City jsou generovány ze statistik, které si tato třída o hře průběžně ukládá.

Ve třídě *SpeechClass* se generuje jeden z projevů emocí. Vybírají se zde textové komentáře (tzv. hlášky), kterými bot komentuje dění ve hře. Hlášky jsou vybírány na základě aktuálních emocí.

TimerClass obsahuje množství různých časovačů, které se používají při

vykonávání rozličných akcí (aby tyto akce mohly být prováděny správně). Jsou to například časovače generování statistik, časovače kontroly, jestli se bot pohybuje, apod.

Třídy *CombatInfoClass* a *PositionInfoClass* sdružují informace o vybraných předmětech, protivnících a navigačních bodech. Tyto informace jsou využívány dále především behaviorálními moduly.

Třída *utilities* obsahuje pomocné funkce sloužící různým účelům (práce s vektory apod.).

Více informací o implementaci projektu UT Emotion Bots naleznete v dokumentaci [17].

Behaviorální moduly

Behaviorální moduly jsou součástí POSH plánu (definují akty a smysly). Každý behaviorální modul sdružuje smysly a akty, které se vážou k určitému chování – např. pohyb, boj apod. Projekt obsahuje tyto behaviorální moduly (podrobnosti viz. dokumentace [17]):

- *combat_behavior* – sdružuje smysly a akty používané v bojovém chování
- *movement_behavior* – obsahuje smysly a akty, které slouží chování pro pohyb
- *emotion_behavior* – obsahuje smysly a akty relevantní k emocím
- *universal_behavior* – sdružuje ostatní smysly a akty, které nešlo zařadit do předchozích modulů

7.1 Emoční model

V této kapitole bude popsána implementace emočního modelu v projektu UT Emotion Bots. Budou zde popsány vnitřní struktury sloužící pro uchovávání pocitů, emocí a citů. Kromě těchto struktur zde také bude popsána vrstva, ve které probíhají emoční reakce a která spojuje pocity a city s emocemi (*SensationsClass*).

Třída emoce

Pro uchovávání emocí slouží v mé implementaci emočního modelu třída *emoce*. Jedna třída emoce reprezentuje jeden komplementární pár emocí. Třída *emoce* je vlastně konečný automat obsahující rovnovážný stav a dva excitované stavy pro každou z komplementárních emocí. Celkem tedy obsahuje pět stavů. Jednotlivé stavy jsou reprezentovány celočíselnou proměnnou *status,* která nabývá hodnot od -2 do 2. Číslo 0 reprezentuje rovnovážný stav. Záporné číslo znamená zvýšenou „dolní“ emoci komplementárního páru, kladné číslo znamená zvýšenou „horní“ emoci. Kromě proměnné status obsahuje třída *emoce* ještě textovou proměnnou *emotion*, ve které je uložen název momentální excitované emoce z páru. V případě, že není momentálně excitována ani jedna emoce z páru, je hodnota nastavena na „neutral“.

Přechody mezi stavy jsou činěny s jistou dávkou náhody, platí ovšem, že čím

víc je jedna z emocí stimulována, tím se zvyšuje pravděpodobnost přechodu do vyššího stavu dané emoce. Implementačně se pro tento systém používá proměnná *value*, která je součástí třídy. Proměnná *value* může dosahovat hodnot od -1 do 1 (real). Čím vyšší je absolutní hodnota proměnné *value*, tím je vyšší pravděpodobnost, že se přejde při příští stimulaci emoce o jeden stav nahoru nebo dolů. (kladné číslo – šance přejít nahoru, záporné dolů). Vzorec používaný při stimulaci emoce pro výpočet přechodu do vyššího stavu emoce vypadá následovně (pro „dolní“ emoci jsou všechna znaménka opačná):

```
value = value + jump;
if value + random(0,1 to 0,5) > 0,9 then
    increase emotion;
    value = + 0,1;
```

Proměnná *jump* reprezentuje hodnotu, o kterou se proměnná *value* při každé stimulaci zvýší (sníží). Proměnná *jump* může být implicitně nastavena na určitou hodnotu a lze pomocí ní ovlivnit pravděpodobnost, s jakou stimulovaná emoce přejde do vyššího (nižšího) stavu.

Speciální případ nastává, pokud je emoce už ve svém nejvyšším excitovaném stavu (-2 nebo +2) a mělo by dojít k dalšímu přechodu směrem nahoru. V takovém případě k přechodu nedojde a je nastavena hodnota proměnné *excited* na -1, resp. +1. V tomto případě jsou ignorovány všechny další stimulace takto postižené emoce, dokud se proměnná *excited* nezmění.

Emoční model obsahuje 8 emocí ve 4 komplementárních párech. Pro reprezentaci všech emocí jsou nutné 4 třídy emoce. Informace o tom, které emoce bude jedna třída reprezentovat, obdrží třída při své konstrukci.

Třída SensationsClass (sensations)

Třída *sensations* je hlavní vrstvou emočního modelu. V této třídě jsou reprezentovány pocity a city a zde také probíhá interakce mezi nimi a emocemi. Také zde probíhá přímá reakce na podněty z prostředí pomocí pocitů. Součástí této třídy jsou i metody určené pro generování statistických údajů ze hry. Tyto údaje se využívají při generování citů k ostatní hráčům ve hře.

Vyhodnocování zpráv z prostředí je prováděno prostřednictvím pocitů. Jeden pocit je reprezentován jednou reálnou proměnnou. Vždy při příchodu nějaké emočně relevantní zprávy z prostředí je zavolána metoda ve třídě *sensations*, která je s touto zprávou asociována, a v ní je následně proveden výpočet, jaké pocity s jakou hodnotou se vygenerují. Pokud se zpráva nějakým způsobem týká jiného hráče ve hře, pak se při výpočtu generovaných pocitů vezme do úvahy cit, který je s tímto hráčem asociován. Asociovaný cit může zásadně ovlivnit množinu pocitů, které budou na základě zprávy z prostředí vygenerovány.

City jsou uloženy současně se statistikami ve struktuře *feelings* (*feelings* je pythonovský slovník). Klíčem v této struktuře je ID hráče. V rámci statistik se ukládají údaje jako jméno hráče, počet zabití a smrtí, počet střel, jež byly zaznamenány jako úspěšné z obou stran, momentální cit, který bot ke hráči chová (cit

je uložen v textové podobě) a údaje o posledním setkání s daným hráčem a o četnosti setkání. Speciální je položka „Attraction“, která je náhodně vygenerována při prvním setkání s daným hráčem a která reprezentuje jakousi implicitní náklonnost našeho bota k danému hráči. Tato položka pak hraje významnou roli při výpočtu citu, který bude s daným hráčem asociován.

Výpočet aktuálního citu, který bude asociován s daným hráčem, se uskutečňuje vždy při změně nějakého údaje ve statistikách. Probíhá tak, že se na základě statistik vypočítá několik koeficientů (například poměr smrtí způsobených daným hráčem s počtem zabití daného hráče), tyto koeficienty se porovnají mezi sebou a s položkou „Attraction“ a výsledkem je nový aktuální cit (více v dokumentaci [17]).

Nyní se dostáváme k jádru interakce pocity – emoce. Aktuální pocity jsou zpracovávány jednou za sekundu metodou *generate_emotions*. V této metodě se rozhoduje, které emoce budou stimulovány. Metoda porovnává hodnoty pocitů. Vždy dva pocity, překračující určitou mezní hodnotu, jsou nutné pro stimulaci emoce, která je k nim asociována. Následuje seznam přechodových rovnic:

```
if rozkoš > 1 and očekávání > 1 then stimulace pýchy
if bolest > 1 and zmatenost > 1 then stimulace studu
if zmatenost > 1 and odpor > 1 then stimulace strachu
if bolest > 1 and překvapení > 1 then stimulace hněvu
if odpor > 1 and zmatenost > 1 then stimulace smutku
if rozkoš > 1 and přitažlivost > 1 then stimulace radosti
if objevení > 1 and překvapení > 1 then stimulace pobavenosti
if očekávání > 1 and zmatenost > 1 then stimulace utahanosti
```

Na tento seznam se můžeme dívat jako na pravidla pro stimulaci emocí. V jednom vyhodnocování jsou zkontrolována všechna tato pravidla. Jeden pocit, překračující mezní hodnotu, může stimulovat více emocí. Pocity, které vedly ke stimulaci nějaké emoce, jsou nastaveny na hodnotu nula. Ostatní pocity mohou určitou dobu přetrvat, dokud nejsou využity při další stimulaci, nebo dokud nejsou pročištěny jako staré (mazání všech pocitů probíhá jednou za několik sekund).

Shrnutí

Emoční model obsahuje čtyři třídy *emoce* uchovávající emoční stav bota a jednu třídu *sensations* zprostředkovávající okamžité reakce na prostředí. V této třídě se dále nachází pocity a slovník *feelings*, ve kterém jsou uloženy city.

Při vyhodnocování podnětů hrají důležitou roli city. Emoce jsou stimulovány na základě pocitů a výpočet pro stimulaci probíhá jednou za sekundu. Pocity jsou reprezentovány jako reálná čísla, city jsou reprezentovány ve statistikách jako textové proměnné a emoce jsou reprezentovány jako stavové automaty třídou *emoce*.

7.2 Projevy emocí

V této kapitole popíšu implementaci emočních projevů bota vyjma emočního chování, které bylo popsáno dříve. Bude se tedy jednat o implementaci komentování dění ve hře (speech systému) a popis změn a rozšíření, které bylo nutné učinit v rozhraní GAMEBOTs, aby bylo možné využívat gestikulaci a ovládání rychlosti bota ve hře.

7.2.1 Třída SpeechClass

Tato třída se stará o komentování dění ve hře prostřednictvím textových zpráv zasílaných do textového kanálu hry UT. Pro odesílání zpráv se využívá metoda třídy *UT_Bot send_global_message*. Třída *SpeechClass* obsahuje množství časovačů, které slouží k tomu, aby zprávy nebyly odesílány příliš často. Ke každé zprávě se vážou dva časovače. *Next_<jméno_zprávy>_message* udává čas, odkdy může být další zpráva daného typu odeslána a *<jméno_zprávy>_timer* udává položku, která je využívána při výpočtu hodnoty prvního časovače. Pro výpočet hodnoty *Next_<jméno_zprávy>_message* se používá následující vzorec:

```
Next_<jméno_zprávy>_message = čas_odeslání_zprávy +
<jméno_zprávy>_timer + random(0 to <jméno_zprávy>_timer)
```

Třída obsahuje také proměnné *minimum_diference* a *last_message_time,* které slouží k tomu, aby nebyly odeslány dvě zprávy bezprostředně za sebou. Hodnota proměnné *minimum_diference* specifikuje nejmenší možný časový rozestup dvou různých zpráv.

Všechny zprávy jsou podle svého typu uloženy v pythonovském slovníku *speech_dict*. Klíčem slovníku je vždy textový název typu zprávy, který se skládá z emoce, z úrovně excitace dané emoce a z akce, ke které se zpráva váže (například *anger_greet_lvl1* znamená, že se jedná o zprávu asociovanou s emocí hněv, excitovanou na úroveň jedna a používanou jako pozdrav ostatním hráčů). Jako hodnota klíče ve slovníku je potom pythonovský seznam řetězců *string*, které reprezentují možné zprávy pro danou událost, emoci a její úroveň excitace.

Abychom mohli vybrat správnou zprávu, musíme nejdříve určit emoci, případně jednu z emocí, které má bot excitované. K těmto účelům slouží metoda *determine_emotion*. Metoda *determine_emotion* vybírá emoce podle úrovně excitace. Při shodě je priorita výběru následující: Nejvyšší prioritu má dvojice emocí hněv a strach, následuje pýcha, stud, radost a smutek a nejnižší prioritu mají emoce pobavenost a utahanost. V případě, že není excitována ani jedna z emocí, je emoční stav vyhodnocen jako neutrální (neutral).

7.2.2 Rozšíření GAMEBOTs

Při implementaci GAMEBOTs byl kladen důraz na pohyb bota po herní lokaci a zprostředkovávání informací z prostředí. Nebyly implementovány žádné funkce podporující zobrazování emocí. Tyto funkce bylo nutné implementovat, stejně jako bylo nutné opravit některé další chyby, které ztěžovaly nejen projevy

emocí, ale i samotné ovládání bota.

Modifikace kódu GAMEBOTs, které jsem učinil v rámci projektu UT Emotion Bots se dají shrnout do tří kategorií. Zaprvé opravení některých chyb a překlepů v kódu GAMEBOTs, zadruhé přidání dalších parametrů v rámci existujících zpráv a příkazů v GAMEBOTs a zatřetí vytvoření nových příkazů umožňujících další způsoby vyjadřování emocí ve hře (což zahrnuje i příkazy umožňující přesnější ovládání bota). Dále uvedu jen nejdůležitější opravené chyby a přidané příkazy.

Opravené chyby v GAMEBOTs

- *opraven příkaz VMT* – byl opraven příkaz sloužící k posílání textových zpráv ve hře v rámci jednoho týmu (tak, že druhý tým zprávy nevidí).
- *opravena chyba při nastavování jména bota* – nyní je možné nastavit, jaké bude mít bot ve hře jméno. Předtím byl tento parametr ignorován.
- *opraven příkaz JUMP* – opravena chyba, která umožňovala opakovaným voláním příkazu JUMP vyskákat až ke stropu.
- *opravena zpráva PLR* – zpráva PLR obsahující informace o viditelných hráčích nyní posílá i jejich jména.

Nové příkazy a zprávy v GAMEBOTs

- *příkaz ANIMATE* – nový příkaz v GAMEBOTs umožňující spouštění nativních gest přítomných ve hře UT. Momentálně jsou k dispozici gesta wave (mávání) a victory dance (vítězný tanec).
- *příkaz SETSPEED* – nový příkaz umožňující přesné ovládání rychlosti bota ve hře.
- *příkaz TRACE a FTRACE* – nové příkazy TRACE a FTRACE umožňují získat dodatečné informace o topologii mapy pomocí paprsku vyslaného mezi dvěma místy. Součástí příkazů jsou zprávy TRC a FTR, které vrátí výsledek letu paprsku (jestli prošel aniž by do něčeho narazil, případně souřadnice místa, kde narazil paprsek na překážku).
- *zpráva SPW* – zpráva by měla přijít ve chvíli, kdy je bot po smrti znova vytvořen („respawnován“) někde na mapě.

8 Zhodnocení

V této kapitole zhodnotím, nakolik se podařilo vytvořit uvěřitelné protivníky a také jakou měrou byla uvěřitelnost ovlivněna jednotlivými platformami a jednotlivými aspekty návrhu bota (emoční model, projevy emocí apod.). Při hodnocení je důležité si uvědomit, že UT není skutečný svět. Je to akční 3D hra a uvěřitelné chování zde tedy může zahrnovat některé aspekty, které se ve skutečném světě vymykají normálu (střelba a eliminace protihráčů je normální součástí hry).

Dále bude v této kapitole hodnocena má implementace Champandardova emočního modelu, na kterou jsem při práci kladl největší důraz. V rámci implementování emočního modelu jsem získal mnoho zkušeností a učinil několik poznatků, které vytvářejí dobrý základ pro další práci.

Na závěr této kapitoly zhodnotím platformy POSH, UT a GAMEBOTs z hlediska jejich vhodnosti pro vývoj umělé inteligence v FPS hrách.

8.1 Uvěřitelnost

Problém uvěřitelnosti se při implementaci emočních botů ukázal být mnohem komplexnější než jsem očekával. Ukázalo se, že kvalitní emoční model je jen jedna ze součástí, která ovlivňuje uvěřitelnost protivníků. Přesto jsem se při implementaci rozhodl věnovat emočnímu modelu nejvíce času, abych získal co nejvíce poznatků zejména z oblasti, jak může emoční model ovlivňovat botovo chování. Zjistil jsem totiž, že uvěřitelnost není jediné možné pole působnosti emočního modelu.

Rozhodnutí více se věnovat emočnímu modelu (tak jak jsem původně zamýšlel) mělo za následek, že jsem strávil adekvátně méně času při tvorbě chování bota, které nemalou měrou přispívají k celkové uvěřitelnosti. Největší problémy bota z hlediska uvěřitelnosti se dají shrnout do tří skupin – pohyb po mapě, platforma GAMEBOTs a přehled o prostředí.

Pohyb po mapě

Problém uvěřitelnosti bota je nejvíce patrný při pohybu po mapě. Náš bot – na rozdíl od originálních UT botů – vstupuje na mapu bez jakýchkoli znalostí o poloze předmětů či o topologii mapy. Toto je výrazný handicap zejména kvůli tomu, že reprezentace mapy v UT se omezuje pouze na síť way pointů. Bot musí začít jen na základě takto kusých informací mapu nějakým způsobem prozkoumávat, což nevede k velmi uvěřitelnému chování. Pohyb po mapě bude zlepšen začleněním výsledků projektu UT NavMesh Jakuba Gemrota, do té doby jsem se rozhodl řešit tento problém elementárně (bot prozkoumává mapu náhodně).

Platforma GAMEBOTs

Dalším z problémů, které nepřispívají k větší uvěřitelnosti bota, jsou některé nedodělky platformy GAMEBOTs. Zejména se to týká spouštění animací – animace běhu, chůze apod. Občas se totiž stane, že ačkoli bot stojí na místě nebo se jen otáčí, tak je stále spuštěna animace běhu. Toto je čistě problém GAMEBOTs, který kazí celkový dojem z bota. Dále GAMEBOTs nikdy nespouštějí animaci otáčení při otáčení. Tyto problémy se, bohužel, ukázaly až při testování bota a celkově musím říci, že činí platformu GAMEBOTs nevhodnou pro vytváření uvěřitelných počítačových protivníků.

Přehled o prostředí

Posledním z problémů, které nevedou k větší uvěřitelnosti bota, je botův malý přehled o dění v prostředí. Občas se stane, že bot přehlédne protivníka, který projde v jeho blízkosti. Problém je částečně způsoben absencí výsledků projektu UT NavMesh a částečně občas méně rychlým vyhodnocováním smyslů v architektuře POSH. Až bude mít bot informace o rozložení prostoru na mapě (projekt UT NavMesh), bude se moci na základě těchto informací rozhlížet směrem, kde je velká

pravděpodobnost, že může někoho zahlédnout.

Na základě uvedených problémů jsem se rozhodl dívat se na uvěřitelnost spíše z pohledu emočního modelu, než z pohledu emočních projevů. Ačkoli momentální projev bota není příliš uvěřitelný, ukázalo se, že emoce mohou uvěřitelnost bota zlepšit. Všechny poznatky, které jsem při práci s modelem získal, shrnu v následujících částech této kapitoly.

8.1.1 Projevy emocí – výkonnost bota, uvěřitelnost a předvídatelnost

V této kapitolu popíšu, jak projevy emocí (a tedy emoce) ovlivňují jednotlivé aspekty botova chování – zejména celkovou výkonnost bota, uvěřitelnost a v souvislosti s ní předvídatelnost botova chování.

Výkonnost bota

Výkonnost bota spíše než samotný emoční model ovlivňují projevy změn emocí, které jsou implementovány. V mém projektu nebyl kladen důraz na výkonnost bota v počítačové hře, co se týče efektivity hraní, ale spíše na odlišení jednotlivých emočních chování. Toto bylo implementováno i za cenu nižšího výkonu bota, což se projevilo nejvíce, když bojoval s originálními boty v UT.

Když jsem bota testoval proti originálním botům z Unreal Tournamentu, měl můj bot velkou nevýhodu, protože v momentě, kdy UT bot začal vyhrávat a vybavil se, byl můj bot na základě toho smutný, přestal se zajímat o dění ve hře, přestal sbírat zbraně, což nakonec vedlo k dalším a k dalším prohrám a k dalšímu prohloubení těchto emocí.

Uvěřitelnost

Při práci se potvrdilo, že emoční model a projevy emocí mají podstatný vliv na uvěřitelnost botova chování. Domnívám se, že projevy emocí mohou zvýšit uvěřitelnost botů v FPS hrách. Opravdu zhodnotit celkový přínos emocí pro uvěřitelnost bota by ale nemělo stát na jednom člověku. Pro účely hodnocení uvěřitelnosti botů projektu UT Emotion Bots jsem žádnou hodnotící skupinu lidí neorganizoval, domnívám se totiž, že by to bylo předčasné. V momentě, kdy jsem si uvědomil důležitost habituace, se kterou se v modelu nepočítalo (bude vysvětleno v kapitole 8.2.2), rozhodl jsem se, že se nebudu pokoušet o další vylepšování uvěřitelnosti. Implementace UT Emotin Bots vedla kromě toho také k několika dalším zjištěním (kapitola 8.2) a dále jsem ji proto využíval jako testovací implementaci, jak může bot emočně reagovat na prostředí a ověřoval jsem na ní některé hypotézy.

Tvořit emoční model a vytvořit uvěřitelné protivníky jsou dva cíle, které spolu sice souvisí, ale jsou samy o sobě poměrně komplexní. Ačkoli se v rámci projektu UT Emotion Bots nepodařilo vytvořit boty s příliš uvěřitelným chováním, podařilo se ukázat některé pozitivní přínosy emocí a cestu, jakou by se člověk měl ubírat, chce-li uvěřitelné protivníky vytvářet.

Předvídatelnost

Předvídatelnost botova chování je nejvíce ovlivněna celkovým počtem jednotlivých různých emočních chování a celkovou komplexností emočního modelu, který mezi těmito chováními na základě událostí ve hře vybírá. Čím je model

komplexnější a čím je počet emočních chování větší, tím se bot stává méně předvídatelným. Bot totiž kromě reakcí na události ve hře – např. vidím nepřítele, reaguje také na své vlastní emoce. Hráč neví, jaké emoce má bot právě excitované, může to zjistit až druhotně z jeho reakcí, které ho v takovém případě mohou překvapit. Emoční model použitý v Emotion botech je poměrně komplexní zejména díky pocitové vrstvě. Přecházení do vyšších stavů emocí s jistou dávkou náhody také pomáhá v modelu snižovat předvídatelnost botova chování.

Vysoká předvídatelnost chování je přitom jedním z problémů botů vyskytujících se ve většině FPS her současnosti. Věřím, že emoční model může být použit jako vrstva zaměřená přímo na snižování předvídatelnosti chování (se zachováním uvěřitelnosti chování). Nízká předvídatelnost může dále zvýšit výkon bota, ovšem jen v kombinaci s projevy chování implementovanými v souladu s tímto požadavkem.

Shrnutí

Zjistil jsem, že pro zvýšení uvěřitelnosti botova chování není nutné implementovat komplexní emoční model, ani velkou množinu různých emočních chování. Místo toho je dobré soustředit se na několik málo emočních chování a tyto od sebe dobře rozlišit. Každé z těchto jednotlivých chování by mělo být dále samo o sobě uvěřitelné. Paradoxně uvěřitelnost chování zvyšují drobné excesy nebo nelogičnosti, které se občas v botově chování mohou objevit. Z hlediska hráče můžou tyto jevy vdechnout botovi zdání vlastní osobnosti. Je nutné udržet tyto jevy na nějaké normální úrovni výskytu – neměly by se vyskytovat příliš často.

8.2 Emoční model

Champandardův model představuje komplexní řešení, které se opírá o poznatky z psychologie, zároveň však bere v úvahu technická omezení platforem, na kterých je model vyvíjen. Model je v knize [2] popisován spíše obecněji, konkrétní implementace je ponechána na programátorovi. Champandardovi šlo zřejmě o to, specifikovat hlavní problémy, které se musí při implementování emočního modelu řešit a navrhnout jedno možné obecné řešení.

Bohužel se mi nepodařilo nalézt žádnou další implementaci Champandardova modelu mimo velmi jednoduché ukázkové implementace (na platformě FEAR, více v [2]), která byla součástí přílohy knihy [2]. Součástí ukázkové implementace bylo jen několik málo přechodů emocí a asi čtyři události v prostředí, na které se reagovalo. Nejzajímavější je asi to, že v této implementaci pracoval Champandard s pocity jako s logickými proměnnými true a false.

Dále rozeberu největší klady a zápory Champandardova modelu v souvislosti s mou implementací.

8.2.1 Problém vyvážení

Ukázalo se, že nejtěžší částí celého návrhu je specifikovat přesně, na jaké události bude bot reagovat a také specifikovat způsob reakce. Zejména se jedná o rozhodnutí, jaké pocity s jakou hodnoutou generovat při přijmutí určité zprávy z prostředí. Obecně se dá říci, že se jedná o vyvážení emočního modelu.

Champandard se zřejmě tento problém snažil zjednodušit pomocí vrstvy pocitů, která hraje v jeho modelu důležitou roli. Věřil, že bude popsání určitých situací v prostředí pomocí nízko úrovňových pocitů intuitivní a jednoznačné. Při implementaci se ukázalo, že toto není zcela pravidlem. Zejména u komplexnějších situací bylo otázkou, jaké pocity se mají vygenerovat. Často jsem jako člověk věděl, jakou emoci bych v dané chvíli očekával. V Champadardově modelu jsem musel ale první generovat pocity. Používal jsem často způsob, že jsem generoval pocity podle toho, jakou emoci bych očekával. Nicméně potom se mi zdálo, že některé události generovaly pocity vcelku nelogicky.

Například mi přišlo přirozené, když je bot zasažen nepřítelem, aby byla potencionálně stimulována emoce hněv. Podle přechodových rovnic jsem tedy v reakci na zásah generoval pocity bolest a překvapení. Proč by ale bot měl být při každém zásahu překvapený? Co když zásah očekává? Pokud bych na druhou stranu generoval pouze pocit bolest, který je v tomto případě logický, pak by zásah sám o sobě nikdy hněv nestimuloval. Mohlo by k tomu dojít pouze ve spojitosti s nějakou další událostí v prostředí a zde je otázkou, zda je to správné.

Champandard také přesně nepopisuje, jak mají být pocity zpracovávány. Tedy jestli se mají z různých zpráv kumulovat například po dobu několika sekund a poté vyhodnotit, nebo zda má být několik zpráv z prostředí (například tři) vyhodnoceno naráz. Lepší specifikace vyhodnocování pocitů by problém vyvážení mohla vyřešit. S vyvažováním modelu souvisí další problém a tím je nulová habituace v modelu.

8.2.2 Problém habituace

Habituace je jev pozorovaný u lidí a zvířat. Jedná se o postupné zvykání si na podněty z prostředí. Nejlépe to ilustrujeme na příkladu. Řekněme, že jsme právě přišli do kanceláře, kde každou chvíli zvoní telefony. Zpočátku nás bude zvonění telefonů rušit a vyvádět z koncentrace. Pokud zazvoní telefon, velmi pravděpodobně zvedneme hlavu a podíváme se, co se děje. Pokud ale v kanceláři strávíme více času, postupně přestaneme zvonění telefonů tolik vnímat. Může to dojít až do stadia, kdy začneme zvonění telefonů úplně ignorovat.

V souvislosti s modelem je problém následující: Některé události v prostředí se mohou velmi často opakovat. Vezměme například znova událost zásahu bota střelou. Taková událost bude v Champandardově modelu neustále generovat ty samé pocity s tou samou hodnotou. Toto může mít za následek v horším případě neustálou stimulaci jedné emoce, v lepším případě bude neustále vysoká hodnota jednoho pocitu, což opět povede k preferenci stimulace určité skupiny emocí (toto závisí na pocitech, které událost generuje). V Champandardově modelu se s habituací vůbec nepočítá, ačkoli je to jeden z jevů, který velmi ovlivňuje lidské chování.

Dalším z problémů, které mohou kvůli absenci habituace nastat, je neschopnost bota přizpůsobit se novému prostředí. Jednotlivé mapy ve hře UT se mohou zásadně lišit (velikost mapy, počet předmětů apod.). To, že je emoční model vyvážený na jedné mapě, neznamená, že musí být vyvážený na všech. Problém může nastat například na mapě obsahující mnoho předmětů, bot se tomu nikdy nepřizpůsobí a bude stále generovat pocit zmatenost.

Díky habituaci by se botovi emoční reakce dříve či později přizpůsobily prostředí mapy. Model by se vlastně pomocí habituace vyvažoval sám za chodu v závislosti na tom, jaké stimuly a jak často by z prostředí přicházely. Celkově by habituace měla vést k větší vyváženosti emočních reakcí bota.

Implementace habituace

Z implementačního hlediska by to mohlo vypadat následovně: Habituaci by zastupovala funkce, která by monitorovala příchozí stimuly z prostředí. Pokud by nějaký stimul přicházel velmi často (například střelba), funkce by v závislosti na tom, jak často tento stimul přichází, zmenšovala hodnoty pocitů, které by tento stimul generoval. Naopak u stimulů, které by se příliš často nevyskytovaly, by byla hodnota generovaných pocitů postupně zvyšována, opět v závislosti na tom, jak dlouho se už neobjevily. Předpokládám, že by bylo nutné nastavit rozmezí, tedy nejnižší a nejvyšší hodnotu pocitů, kterou daný stimul může generovat.

Nevýhody habituace

Předpokládanou nevýhodou habituace by bylo zvětšení nároků emočního modelu, zejména co se týče velikosti zabrané paměti. Musely by se zavést statistiky přijatých stimulů. Zřejmě by se také musel změnit způsob vyvažování modelu, tedy generované implicitní hodnoty pocitů a bylo by nutné stanovit a otestovat nejnižší a nejvyšší možné hodnoty generovaných pocitů. V tomto smyslu by bylo vyvažování modelu složitější. Na druhou stranu možnost modelu nastavovat se za chodu sám, by nakonec mohla vést k menší náročnosti nastavení pro programátora. Pro přesné ověření těchto hypotéz by bylo nutné zkusit model s habituací implementovat.

Momentální řešení problému

Jednodušší ale na druhou stranu pouze částečné řešení problému habituace, které v projektu momentálně používám, je následující: Pokud je v modelu nějaká emoce stimulována velmi často a přejde do nejvyššího stavu a chce se dále zvyšovat, je na určitou dobu zablokována. To znamená, že nemůže být dále zvyšována, ale jen snižována. Čím delší je doba této inhibice, tím je pravděpodobnější, že dojde k postupnému snížení dané emoce a je zde tedy možnost, že převládnou ostatní emoce. Pokud je doba inhibice příliš dlouhá, tak má bot vcelku pravidelně excitované různé emoce a jeho chování je pestré, méně předvídatelné, nicméně nepříliš uvěřitelné (nastává pravidelné cyklování mezi několika emocemi). Optimální je krátká až střední doba inhibice (jedna až deset sekund).

8.2.3 Význam a přínos citů

City jsou jedním ze základních konceptů Champandardova modelu. Při implementaci jsem zjistil, že mají velký vliv na chování bota. Jejich přínos, co se týče emočního modelu, byl velký a je jisté, že se vyplatí věnovat jim zvýšenou pozornost. Získal jsem dokonce dojem, že by bylo vhodné tento koncept oproti Champandardově modelu ještě více rozšířit. Dále vysvětlím, proč takto usuzuji.

City nám umožňují asociovat emoce s objekty v prostředí. Tyto objekty mohou být ostatní postavy, které se ve hře vyskytují, ale i například předměty, které bot používá. Velkou výhodou citů je to, že poskytují emoční hodnocení v delším časovém úseku. Vytvářejí jakýsi trvalejší postoj k objektům ve hře, který je založený na předchozích událostech.

V mé implementaci Champandardova modelu existují 4 city, které mohou být asociovány s ostatními hráči ve hře. City zásadně ovlivňují emoční reakce na události v prostředí a přibližují je více skutečným lidským reakcím. Přitom jejich koncept není nijak složitý – jsou generovány na základě statistik, které jsou o hře a

o jednotlivých hráčích průběžně ukládány. Velmi často je celá komplexnost botova chování založená jen na citu, který má bot zrovna asociován s jiným hráčem ve hře.

Nejvíce patrné je toto asi u citu přitažlivost. Řekněme, že se hráč chová nepřátelsky a zároveň má bot k němu asociovaný cit přitažlivost. Náš bot po takovém hráči sám od sebe střílet nezačne, nicméně v sebeobraně případnou palbu opětuje. Pokud bot v obraně takového hráče zlikviduje, budou se generovat místo obvyklých pozitivních pocitů negativní. Místo toho, aby byl bot na sebe hrdý, že někoho zabil, bude smutný, že způsobil smrt někoho, kdo mu byl sympatický. Hráče může toto chování překvapit – nemusí totiž o žádné vrstvě citů vědět. Následně si může položit otázku – „Proč je ten bot smutný, vždyť mě třikrát zabil...“

I ostatní city ale přispívají k lepšímu chování. Ať už to může být nenávist k někomu, kdo bota stále zabíjí a následná stimulace hněvu, která má za následek bezhlavý útok na takového hráče, nebo celková znechucenost a utahanost, kterou způsobí nějaký hráč, se kterým je asociován cit odpor.

Celkově můžu říci, že city asi největší měrou přispěly k uvěřitelnosti botova chování. Rozšířením citů i na neživé objekty ve hře – například na zbraně – by dle mého názoru vedlo k dalšímu vylepšení emočního modelu.

8.2.4 Emergentní jevy modelu

Při testování modelu jsem byl překvapen některými projevy emočního bota. Tyto emergentní jevy jsou převážně důsledkem celkového konceptu modelu. Občas, naproti mému očekávání, měl bot po své likvidaci excitovanou emoci pýcha. Přitom jsou při úmrtí bota generovány pocity, které by měly generovat právě opačnou emoci stud, smutek a podobně. Ovšem dejme tomu, že bot před smrtí s protivníkem bojoval a několikrát ho trefil – bot je na sebe patřičně hrdý a ačkoli umře, negativní pocity nepřevýší kladné pocity z boje. Tento jev je patrný i u lidských hráčů – například pokud bojují s protivníkem, o kterém si myslí, že je silnější než oni. V takovém případě je i pouhé zranění takového soupeře úspěchem.

Jedním z dalších překvapení, které mě při implementaci emočního modelu čekalo, bylo, že ačkoli jsem zjistil, že vyvážení emočního modelu je poměrně obtížné, může tento model generovat vcelku uvěřitelné chování bez dokonalého vyvážení. Toto mne přimělo uvažovat nad myšlenkou, že pro uvěřitelné chování nemusí být emoční model dokonale vyvážen, mnohdy stačí pouhá jeho přítomnost. Drobné excesy jsou vlastně žádoucí – přispívají k jedinečnosti botova chování. Samozřejmě to platí jen do určité míry. Naprosto nevyvážený model nebude generovat uvěřitelné chování.

8.3 Hodnocení POSH architektury

Jedním z cílů architektury POSH bylo zpřístupnit paradigma BOD za účelem řízení autonomních agentů v různých nehostinných prostředích. Paradigma BOD představuje velmi intuitivní zápis různých druhů chování, který je pro člověka dobře čitelný. Díky intuitivnosti BODu se v POSHi dá napsat velmi rychle a krátce to, nad čím by člověk například v C strávil mnohem více času.

Další výhodou POSHe je rychlost vyhodnocování. Díky stromové struktuře plánu je umožněno rychlé vyhodnocení i poměrně velkých plánů (tomuto napomáhá

ještě slip-stack funkce POSHe).

POSH toho ovšem nabízí mnohem více. Je třeba ale říci, že POSH byl koncipován spíše pro ovládání robotů pohybujících se ve skutečném světě a schopných paralelně provádět více akcí naráz. Některé z vlastností POSHe tak zůstávají v prostředí počítačových her nevyužity. Jedná se zejména o možnost paralelního vyhodnocování více pravidel naráz. Bot v počítačové hře obsahuje většinou jen jedno výpočetní vlákno, možnost paralelismu tedy není možné využít.

Další z těžko využitelných schopností POSHe v počítačových hrách je návrat do přerušeného chování (kompetence). Návrat probíhá následovně: Pokud máme běžící kompetenci, ve které jsme u určitého pravidla a přeruší nás vyšší kompetence, pak se po dokončení vyšší kompetence vracíme zpět do přerušené kompetence na místo posledního vyhodnoceného pravidla. Tato schopnost by mohla být pro bota v počítačové hře dokonce potencionálním zdrojem problémů. Nemáme totiž ani tušení, jaká kompetence nás přeruší a co vykoná – jestli se bot na základě toho přesune na jiné místo apod. Pokračování staré kompetence za těchto podmínek v místě její poslední akce může vést k velmi nedefinovanému chování. Někdo by mohl dokonce namítnout, že tato schopnost popírá čistou reaktivnost POSHovského plánu v tom, že si plán vlastně něco pamatuje. Pro bota v počítačové hře se mi osvědčilo nastavit kompetence tak, aby se při přerušení automaticky restartovaly.

Jednou z nevýhod architektury POSH hlavně pro lidi, kteří s ní začínají pracovat, je absence nějakého kompletního návodu k použití přímo od autorky. Pro česky mluvící je ale k dispozici návod vytvořený naší skupinou v rámci práce na našich projektech [18].

Jedním z námětů na vylepšení POSHe, který vychází z našich zkušeností při práci s ním, je umožnit předávání parametrů smyslům a aktům přímo v POSHi. Nyní musíme nemožnost zmíněného nepohodlně obcházet přes více implementovaných aktů a smyslů. Myslím, že není nutné, aby POSH přímo podporoval unifikaci proměnných. Úplně by stačila možnost posílat smyslům a aktům nějaké konstantní parametry.

Rozšíření pohodlí při práci s architekturou POSH a jejími POSH plány přinese rozhraní ABODE. Toto rozhraní bude sloužit k vizualizaci, editaci a debugování POSH plánu. ABODE je v tomto čase (léto 2006) vyvíjeno a testováno na universitě v Bathu [19].

8.4 Hodnocení rozhraní GAMEBOTs

Když jsem začal pracovat s rozhraním GAMEBOTs, velmi záhy se objevila asi největší nevýhoda tohoto rozhraní – poměrně velký počet chyb a nedodělků (špatné spouštění animací bota apod.). Celková neodladěnost tohoto prostředí nakonec vedla k tomu, že jsem kód GAMEBOTs začal upravovat a chyby odstraňovat.

Nebudu zde zabíhat do detailů, nicméně pro to, aby rozhraní GAMEBOTs opravdu mohlo sloužit jako součást platformy pro vývoj a testování umělé inteligence v Unreal Tournamentu, bude nutné opravit zcela zásadní chybu a tou je občasné samovolné odpojení se bota připojeného pomocí GAMEBOTs. Z pozorování jsme v rámci naší skupiny [18] zjistili, že se tato chyba objevuje více, když je v prostředí přítomno více GAMEBOTs botů. Bohužel se nám tuto chybu nepodařilo

odstranit. Nejsme si ani jistí, jestli se nejedná spíše o vlastnost Unreal Tournamentu. V takovém případě by rozhraní GAMEBOTs muselo svou podobu změnit zásadněji.

Přes všechny nedostatky se ukázalo, že rozhraní GAMEBOTs může být použito pro vývoj komplexní umělé inteligence v prostředí Unreal Tournamentu. Pokud však má být toto rozhraní využíváno i mimo akademickou sféru, bude nutné jej dále upravit.

8.5 Hodnocení frameworku Pogamut

Hlavní funkcí frameworku je extrahovat informace z GAMEBOTs zpráv a umožňovat jednoduchou kontrolu nad botem. Je to vlastně taková pythonovká nadstavba nad GAMEBOTs, která přináší významné zrychlení a ulehčení práce při ovládání a ověřování stavu bota.

Součástí frameworku je i GUI, které umožňuje debugování POSH plánů a GAMEBOTs zpráv. Toto debugování je zatím na nízké úrovni výpisů všech zpráv přicházejících z prostředí a do prostředí, nicméně je to velký pokrok oproti předchozím verzím, kde nebyla možnost debugování žádná.

Framework Pogamut obsahuje velmi komplexní struktury pro ukládání dat získaných z GAMEBOTs a účinné funkce schopné nad těmito strukturami pracovat. Věci jako historie zpráv a herní statistiky jsou samozřejmostí.

Cílem tohoto frameworku je co největší ulehčení práce s platformou POSH, GAMEBOTs, UT pro začínajícího programátora. Při použití frameworku nemusí programátor znát syntax příkazů pro GAMEBOTs, pokud bude používat metody třídy UT_Bot.

Při práci na mém projektu jsem využil jen malé procento schopností frameworku Pogamut, ale i to mi ušetřilo spoustu času při psaní kódu pro vyhodnocování GAMEBOTs zpráv. Myslím, že existence frameworku Pogamut je jedním z důvodů, proč bychom se měli dále zabývat vývojem platformy POSH, GAMEBOTs, UT.

9 Future works

Kromě pokračování ve vývoji emočního modelu (kap. 9.2) a úpravy některých platforem, se kterými pracujeme (kap. 9.1), uvažujeme v rámci naší skupiny [18] o společném projektu UT Capture the Flag Bot Team (UT CTF Bot Team). Cílem tohoto projektu by bylo vytvoření spolupracujícího týmu botů schopných hrát CTF[4] mód hry UT. Ukazuje se ale, že dříve než budeme moci zahájit práci na tomto projektu, budeme muset v rámci naší platformy lépe vyřešit některé základní problémy umělé inteligence v FPS hrách. Jedná se zejména o lepší navigaci a pohyb botů po herní lokaci (tento problém částečně řeší UT NavMesh Jakuba Gemrota [7], ještě je ale nutné vytvořit efektivní struktury, které budou schopny nad NavMeshem pracovat).

4 CTF – Caputre The Flag, v tomto módu hry jsou hráči rozděleni na dva týmy. Každý tým má svou základnu a v ní vlajku. Cílem každého týmu je ukořistit vlajku protivníka a zároveň ubránit tu svou.

9.1 GAMEBOTs + Unreal Tournament

Budoucí úpravy rozhraní GAMEBOTs budou kromě opravení chyb zahrnovat také celkovou optimalizaci rozhraní. Jedná se zejména o nadbytečnost zasílání některých zpráv (například seznam viditelných předmětů je zasílán 10 krát za sekundu). Další optimalizace by mohla zahrnovat zmenšení komunikační režie umožněním zasílat některé parametry v binární podobě. Začlenění těchto úprav do rozhraní by mělo za následek změnu celé koncepce GAMEBOTs v tom, jak získává informace o prostředí.

Změna, která se týká platformy Unreal Tournament, zahrnuje zejména otázku přechodu na vyšší verzi této hry (UT 2003, UT 2004). Nové verze UT nabízejí lepší grafické zpracování, větší odladěnost chyb a další vylepšení, přičemž zachovávají vysokou rozšiřitelnost prostředí díky jazyku UnrealScript a editoru UnrealEd, jejichž funkce a podoba zůstává stále stejná.

9.2 Návrh efektivnější implementace emočního modelu

Ve svých budoucích pracích bych rád dále modifikoval Champandardův emoční model a pokusil se lépe navrhnout vrstvu s pocity, lépe specifikovat přechody mezi emocemi a učinit některé další úpravy.

Obsahem této kapitoly budou návrhy úprav mé implementace emočního modelu, které by model měly učinit více efektivním. Efektivitou zde myslím především celkovou složitost modelu v poměru ke generovanému chování z hlediska předvídatelnosti a uvěřitelnosti.

Vypuštění vrstvy pocitů

První zjednodušení by zahrnovalo úplné vypuštění vrstvy s pocity, bot by v reakci na události v prostředí generoval přímo emoce. V souvislosti s tímto bych dále omezil množinu událostí, na které bot reaguje. Emoční reakce by zde zahrnovaly jen zcela zásadní stimuly, například botovu smrt. Stimuly, jako například jestli bot zasáhl nepřítele nebo ne, by byly zcela vynechány. Cílem tohoto zjednodušení je zejména snazší a intuitivnější vyvažování a nastavování modelu.

Začlenění habituace

Součástí upraveného modelu by byla i schopnost habituace. Hodnota emocí, která by byla generována v reakci na události v prostředí by tímto byla ovlivněna. Díky tomu získá model za cenu udržování si dalších statistik schopnost upravovat se za běhu. Habituace by přispěla k větší stabilitě botova emočního stavu.

Rozšíření konceptu citů

City budou nadále ovlivňovat generované emoce. Kromě citů asociovaných k hráčům by model mohl eventuelně obsahovat i city asociované k neživým předmětům ve hře. Například bot má rád zbraň, se kterou zlikvidoval spoustu protivníků. V rámci uvěřitelnějšího chování by model mohl obsahovat větší počet citů a generování citů by mohlo být zpřesněno rozšířením statistik, které se průběžně o hře ukládají.

Začlenění náhodných prvků

Každý člověk je jedinečný, co se týče emočních reakcí. Abychom mohli tento fenomén alespoň zčásti simulovat v emočním botovi, je vhodné začlenit do emočních reakcí jistý náhodný prvek. Botovi bude umožněno chovat se ve stejných situacích různě a otevřeme tím i možnost emergentním emočním jevům – tedy jevům vypadajícím velmi sofistikovaně, nicméně vycházejících z tohoto jednoduchého konceptu. Tento náhodný prvek může být na různých vrstvách a nemusí se vždy projevit. Jedna z vhodných vrstev je ovlivnění hodnoty zvýšení emoce při určité události. Možná ještě lepší využití je generovat emoční reakce na události v prostředí pouze s určitou pravděpodobností.

V modelu by měla být i možnost samovolného přechodu do určité (ale stále uvěřitelné) emoce i pokud by příliš mnoho stimulů z prostředí nepřicházelo – bot je například v lokaci sám. Bot by měl tedy reagovat i na nedostatek stimulů z prostředí. Tyto emoce by ale měly spadat do jiné kategorie než např. strach a hněv. Mělo by se jednat o méně důležité emoce – například pobavenost a utahanost.

Shrnutí

V této kapitole navrhované změny mohou přispět k větší efektivitě emočního modelu. Není nutné a snad ani žádoucí implementovat všechny navrhnuté úpravy v jednom modelu. Záleží na konkrétní situaci a na problémech, které chceme pomocí emočního modelu řešit – uvěřitelnost, předvídatelnost apod.

10 Závěr

Tato práce se zabývala problematikou umělých emocí v počítačových hrách typu FPS. Byl implementován emoční model v projektu UT Emotion Bots v prostředí FPS hry Unreal Tournament. Tento model vycházel z emočního modelu Alexe J. Champandarda, který byl představen v knize [2]. Projekt UT Emotion Bots ukázal velký potenciál emočních modelů pro řešení mnoha problémů počítačových protivníků v FPS hrách současnosti (zvýšení uvěřitelnosti apod.).

V rámci projektu UT Emotion Bots se sice nepodařilo vytvořit dokonale uvěřitelné počítačové protivníky, nicméně se podařilo lépe stanovit faktory, které uvěřitelnost ovlivňují, a také vztah, který má emoční model k těmto faktorům. Selhání z hlediska uvěřitelnosti protivníků nastalo zejména kvůli tomu, že ji kromě emočního modelu zásadně ovlivňují také použité platformy. Ukázalo se, že některé z použitých platforem nejsou pro vytvoření uvěřitelných protivníků vhodné.

Ve svých následujících pracích plánuji dále pokračovat ve vývoji emočního modelu. Pokusím se postupně začleňovat do emočního modelu další poznatky z oblasti psychologie, neboť na implementovaném emočním modelu se ukázalo, že alespoň částečné přiblížení emočního modelu fungování emocí ve člověku přináší velmi nadějné výsledky. Popsal jsem také další zjištěné možnosti využití emočního modelu a navrhl několik způsobů, jak emoční model zefektivnit s přihlédnutím k jeho plánovanému využití.

Věřím, že se na akademické půdě bude zvyšovat zájem o problematiku umělé inteligence v počítačových hrách, neboť se domnívám, že počítačové hry jsou ideální platforma pro testování nových přístupů k tvorbě umělé inteligence. Je to především díky variabilitě jejich prostředí a jejich celkové dostupnosti.

11 Literatura

[1] Epic MegaGames: Unreal Tournament, http://www.unrealtournament.com, [9. 4. 2006].

[2] Alex J. Champandard: *AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors*. New Riders, 2003.

[3] POSH, http://www.cs.bath.ac.uk/~jjb/, [9. 4. 2006].

[4] Gamebots, http://www.planetunreal.com/gamebots, [9. 4.2006].

[5] Pogamut, http://carolina.mff.cuni.cz/~gib/wiki/, [9. 8. 2006].

[6] Ondřej Burkert, *Spolupracující dvojice do Unreal Tournamentu*, bakalářská práce na Matematicko-fyzikální fakultě Univerzity Karlovy v Praze, 2006

[7] Jakub Gemrot, *Mapování, zpracování a reprezentace map pro Unreal Tournament boty*, bakalářská práce na Matematicko-fyzikální fakultě Univerzity Karlovy v Praze, 2006

[8] M. Bída, C. Brom, O. Burket, J. Gemrot: Pogamut – platforma pro prototypování botů v UT. In: *Sborník příspěvků Kognice a Umělý život VI*, Česká republika, Třešť, 2006

[9] Leonard Berkowitz, *Causes and Consequences of feelings.* Cambridge University Press, 2000

[10] Python, 2003, http://www.python.org, [9.4.2006]

[11] Unreal Engine, http://www.unrealtechnology.com, [5. 4. 2006].

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

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

[14] Rodney A. Brooks: A robust layered control system for a mobile robot. In: *IEEE Journal of Robotics and Automation*, 1986.

[15] Jakub Gemrot, Michal Bída: *Pogamut – platforma pro prototypování botů v UT*, přednáška na koferenci Transistor, Česká republika, Praha, 2006

[16] R. Pluchick: Emotion, *A Psychovolutionary Synthesi*. Harper and Row, New York, 1980.

[17] Michal Bída, *Emoční boti v prostředí Unreal Tournamentu – dokumentace*, příloha bakalářská práce *Emoční boti v prostředí Unreal Tournamentu* na Matematicko-fyzikální fakultě Univerzity Karlovy v Praze, 2006

[18] Group Of Inteligent Behaviors in Prague, http://carolina.mff.cuni.cz/~gib, [9. 4. 2006].

[19] ABODE - Advanced Behavior Oriented Design Environment, http://www.cs.bath.ac.uk/~jjb/web/BOD/abode.html, [30. 7. 2006]