VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

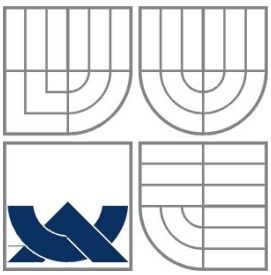
ZADÁVANIE A PLNENIE ÚLOH V 3D PRIESTORE PRE VIACERÝCH UŽÍVATEĽOV

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

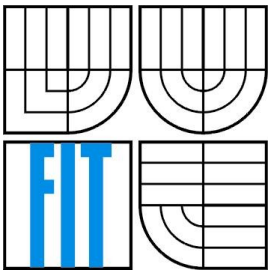
AUTOR PRÁCE
AUTHOR

BORIS KUDLAČ

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZADÁVANÍ A PLNĚNÍ ÚKOLŮ V 3D PROSTORU PRO VÍCE UŽIVATELŮ

SETTING AND COMPLETING OBJECTIVES IN 3D MULTIPLAYER SPACE

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

BORIS KUDLAČ

VEDOUCÍ PRÁCE
SUPERVISOR

Ph.D., Ing. ADAM HEROUT

BRNO 2007

Abstrakt

Cílem této práce bylo vytvoření systému pro zadávaní a plnení úkolů v 3D prostoru pro více uživatelů, konkrétne pro projekt TankGame. Tento projekt je simulací boje tanků a práce do nej přidává možnost zadávat a detekovat libovolné úkoly pro hru jednoho i více hráčů. Zároveň demonstruje tyto možnosti na konkrétním příklade, implementováním soubojového módu hry pro více hráčů.

Klíčová slova

zadávání, plnení, systém, úkoly, manažer událostí, hra pro více hráčů, TankGame, API, simulace, soubojový systém

Abstract

The goal of this thesis is to create system for setting and completing objectives in 3D multiplayer space, in particular for project TankGame. This project is a simulation of tank battles and thesis adds a new feature to it, which is setting and detecting any objectives for single and multiplayer game. In the same time, it demonstrates its capabilities by implementing a deathmatch game for 2 or more players.

Keywords

setting, completing, systém, objectives, event manager, multiplayer, TankGame, API, simulation, deathmatch

Citace

Boris Kudlač: Zadávanie a plnenie úloh v 3D priestore pre viacerých užívateľov, bakalářská práce, Brno, FIT VUT v Brně, 2007

Zadávanie a plnenie úloh v 3D priestore pre viacerých užívateľov

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ph.D., Ing. Adama Herouta.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Boris Kudlač
15.5.2007

Poděkování

Děkuji vedoucímu diplomové práce Ph.D., Ing. Adamu Heroutovi za užitečnou pomoc a cenné rady při zpracování této bakalářské práce.

Obsah

1 Úvod

Táto bakalárska práca nadväzuje na semestrálny projekt. Mala by viesť k získaniu základných poznatkov o spracovávaní a vyhodnocovaní úloh v distribuovanej simulácii, ktorou bol pre túto prácu projekt TankGame. Pri zisťovaní a spracovávaní informácií o riešenom probléme boli využité teoretické poznatky, ktoré sú sústredené v druhej kapitole.

Problém riešený v tejto práci je možno rozdeliť do dvoch menších podproblémov. Prvým z nich je návrh systému pre spracovanie a vyhodnocovanie úloh v simulácii, resp. v projekte TankGame. Keďže sa jedná o simuláciu distribuovanú, zúčastňuje sa na nej teda viacero užívateľov, je treba zaistiť správnu komunikáciu medzi jednotlivými účastníkmi simulácie, čo je podproblémom druhým.

Druhá kapitola pojednáva o spracovávaní a vyhodnocovaní úloh v rôznych aplikáciách. Jej druhá časť je potom zameraná na tento problém v prostredí počítačových hier. Odlíšia sa tu dva typy udalostí a ukážu sa rôzne návrhy riešenia problému. Keďže projekt TankGame možno považovať za sieťovú počítačovú hru, je najväčšia časť kapitoly zameraná na sieťové počítačové hry a problém spracúvania a vyhodnocovania úloh v nich. Sú tu načrtnuté znaky sieťových počítačových hier a ich odlišnosti vo vývoji od počítačových hier pre jedného hráča. Objasní sa tu napr. pojem sieťových udalostí, správcu sieťových udalostí, tabuľky hráčov alebo zdieľaného objektu. Sú tu ukázané dva spôsoby distribúcie informácií v sieťových počítačových hrách. Jedným z nich je všesmerové vysielanie, druhým potom protokol DIS, ktorého princípy sú využité aj v projekte TankGame.

V tretej kapitole sa dozviete najprv všeobecné informácie o projekte TankGame a jeho časti pre viacerých hráčov (multi-player), ako aj konkrétne informácie o realizovaní a implementácii systému pre zadávanie a plnenie úloh, ktorému je projekt TankGame základom. Je tu ukázané fungovanie systému pre všeobecný typ hry a potom demonštrovaná funkčnosť systému na hre pre viacerých hráčov „deathmatch“. Nasleduje popis aplikačného rozhrania systému, konkrétne metód a atribútov triedy `TMatch`, ktorá implementuje ľubovoľnú hru s detekciou úloh, a popis triedy `TObjective`, ktorá reprezentuje bázovú triedu pre ľubovoľnú úlohu. Ďalšia časť kapitoly sa zaoberá komunikáciou v systéme pre zadávanie a detekciu úloh. Tvorí ju popis použitých správ v systéme, ktoré sa využívajú na komunikáciu medzi jednotlivými entitami v distribuovanej simulácii. Záver kapitoly patrí popisu zmien v projekte TankGame. Všetko, čo bolo potrebné na stávajúcom kóde projektu zmeniť alebo pridať, je popísané na tomto mieste.

Záver tejto práce obsahuje zhodnotenie dosiahnutých výsledkov. Popisuje postup tvorby systému pre zadávanie a detekciu úloh v distribuovanej simulácii, jednotlivé stupne návrhu tohto systému a vyriešenie komunikácie entít v systéme. Nakoniec je načrtnuté možné riešenie na ďalší rozvoj tohto systému.

2 Teoretický úvod

2.1 Spracovanie a vyhodnocovanie úloh

V praxi sa stretávame so systémami pre spracovanie a vyhodnocovanie úloh pomerne často, aj keď to tak na prvý pohľad nemusí vyzerať. Jedná sa o vnútorné systémy aplikácií, ktoré spracúvajú vonkajšie alebo vnútorné udalosti a starajú sa o vykonávanie k nim prislúchajúcich funkcií. Stretneme sa s nimi napr. pri všetkých aplikáciách s užívateľským rozhraním, kde tieto systémy riadia väčšinou vonkajšie zásahy užívateľa, a vykonávajú činnosti, ktoré sa od aplikácie po týchto zásahoch očakávajú. Takýto systém, alebo komponenta programu, sa označuje anglickým termínom e*vent manager* [2], voľne preložené ako *správca udalostí*. Správca udalostí je priamo spojený so *správcom úloh* (anglicky *task manager*), ktorý obsahuje konkrétne reakcie na dané udalosti. Ďalšou oblasťou, kde je možno postrehnúť podobný koncept spracovania a vyhodnocovania úloh, sú počítačové hry. Tu sú zadané určité úlohy, ktoré môže hráč plniť, v aplikácii teda musí existovať komponenta, ktorá dokáže úlohy spracovávať a vyhodnocovať. (Táto bakalárska práca sa týka hlavne tejto oblasti.)

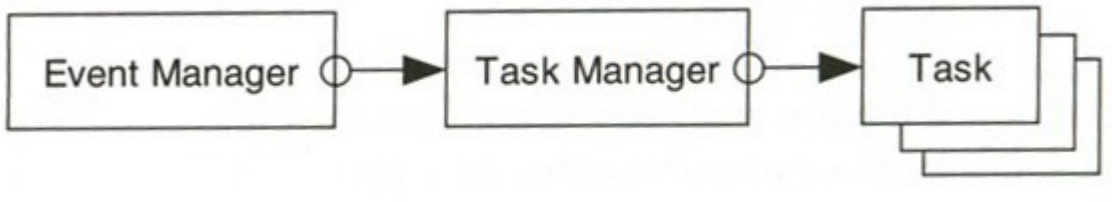
Existuje viacero prístupov k spracovaniu udalostí a systém so správcom udalostí, spojený so správcom úloh, je jedným z nich. Ukážeme si najprv, ako tento systém všeobecne funguje.

2.1.1 Správca udalostí a správca úloh

Funkciou správcu udalostí je detekovanie a prijímanie udalostí od vonkajších alebo vnútorných častí programu. Medzi vonkajšie udalosti patria napr. vstupy užívateľa, v sieťovom prostredí to môže byť prijatie správy od klienta a pod. Vnútornými udalosťami môžu byť naopak rôzne navonok neviditeľné udalosti, ako napr. alarm časovača alebo vnútorná komunikácia medzi objektmi. Akonáhle prijme správca udalostí správu reprezentujúcu udalosť, podľa typu udalosti určí, čo sa bude ďalej vykonávať. Je na jeho implementácii, či reakciu určitý čas pozdrží, alebo reaguje hneď. Ak sa jedná o real-time systémy závislé na dĺžke odozvy, reakcia sa vykonáva zvyčajne bezprostredne po obdržaní správy. V iných aplikáciách môže byť ale reakcia odložená a o jej načasovanie sa môže postarať plánovač. Reakcia sa potom prevedie až v určitom čase, ktorý najviac vyhovuje behu aplikácie.

Pre vlastné vykonanie reakcie musí správca udalostí komunikovať so správcom úloh. Ten obsahuje zoznam úloh, ktoré prislúchajú k jednotlivým udalostiam. Pri rozpoznaní typu udalosti správcom udalostí sa zo správcu úloh vyberie príslušná metóda a prípadne sa aj rovno vykoná.

Takýto postup je najviac používaný v grafických užívateľských prostrediach, ale v obmenenej podobe sa využíva aj pre počítačové hry.



Obrázok 1: Systém so správcom udalostí a správcom úloh

2.2 Spracovanie a vyhodnocovanie úloh v počítačových hrách

V počítačových hrách sa stretneme so spracovaním udalostí a úloh hneď v dvoch podobách. Nachádza sa tu klasický správca udalostí, ktorý prijíma všetky bežné udalosti aplikácie a vyberá k nim príslušné reakcie. Príkladom je stlačenie klávesy pre pohyb vozidla v automobilovej simulácii a následné vykonanie pohybu auta po rozpoznaní vonkajšej udalosti. Hry však disponujú určitými cieľmi, sú riadené danými pravidlami a zadávajú hráčovi rôzne úlohy, ktoré môže splniť. Udalosti týkajúce sa práve týchto úloh, chápaných ako cieľov, tvoria druhú skupinu spravovania úloh. Príkladom môže byť zadanie cieľa v podobe zničenia troch nepriateľských vozidiel a následné odmenenie bodmi v prípade splnenia úlohy.

Existujú teda dva druhy rozpoznávania udalostí – rozpoznávanie jasne definovaných vonkajších a vnútorných udalostí aplikácie a rozpoznávanie vytvorených a zadaných cieľov hry. Problémom teda je, ako zadávať a detekovať druhú skupinu udalostí.

2.2.1 Prevod cieľov na vnútorné udalosti aplikácie

Jedným z riešení môže byť prevod cieľov na vnútorné udalosti aplikácie. Ciele sa rozložia na postupnosť známych udalostí, na ktoré vie aplikácia reagovať. Okrem definovaných reakcií bude treba pridať reakcie špeciálne, ktoré budú detekovať ciele v podobe postupností klasických udalostí. Zadávanie cieľov bude potom otázkou rozloženia určitej úlohy na známe udalosti a vyrobenie detektora postupnosti týchto udalostí. Problém môže nastať, ak niektorý cieľ bude natoľko špecifický, že ho nebude možno rozdeliť na známe udalosti aplikácie. Potom takéto riešenie zlyháva a musí sa pristúpiť k odlišným spôsobom.

2.2.2 Vytvorenie metód na detekciu a zadávanie cieľov

Ak sa ciele nedajú rozložiť na udalosti známe aplikáciou, riešením môže byť vytvorenie vlastných metód na ich detekciu a zadávanie. Vytvorí sa špeciálna metóda na každý typ cieľa zvlášť. Kontrola

splnenia cieľa je potom vykonávaná vnútornými mechanizmami hry. Zadávanie cieľov je realizované pridaním úlohy do zoznamu cieľov. Úloha nepotrebuje žiadny ďalší zvláštny popis, keďže jej definíciou je vlastne popis podmienok splnenia cieľa obsiahnutý v metóde na jej detekciu.

2.2.3 Systém zadávania a detekcie cieľov

Systém pre zadávanie cieľov umožňuje pridať ľubovoľný splniteľný cieľ do hry a následne detekovať jeho splnenie. Je zložený zo zoznamu úloh, a komponenty, ktorá v pravidelných intervaloch kontroluje, či boli úspešne splnené. Ak hráč splní úlohu, odstráni sa úloha zo zoznamu. Ak sa v zozname nenachádza žiadna ďalšia úloha, cieľ hry sa považuje za splnený. Vždy existuje jeden primárny, po ktorého splnení sa hra končí. Ďalej môžu existovať úlohy sekundárne, nepotrebné na úspešné ukončenie hry, ale prinášajúce napr. výhody hráčovi, ktorý ich splní. Úlohy sa môžu skladať z podúloh, a tie zase z ďalších podúloh a pod. Takto sa môžu vytvárať ciele na viacerých úrovniach.

2.3 Spracovanie a vyhodnocovanie úloh v sieťových počítačových hrách

2.3.1 Úvod k sieťovým počítačovým hrám

Sieťové počítačové hry sú počítačové hry, ktoré v nejakej forme umožňujú zdieľanie jedného virtuálneho priestoru viacerými užívateľmi súčasne. Ponúkajú tak nový pohľad na využitie počítačových sietí, hlavne internetu, a stali sa fenoménom posledných rokov. Takmer každá nová vyrobená hra, umožňuje sieťové hranie, absencia tohto prvku sa považuje u väčšiny titulov za veľké mínus. Účastníci *multi-player* hier, ako sa tieto hry anglicky nazývajú, obývajú jeden spoločný interaktívny svet vytvorený počítačom. Virtuálne priestory výrazne rozširujú možnosti hry a znásobujú zážitok a hrateľnosť hry. Okrem toho poskytujú aj priestor pre komunikáciu medzi jednotlivými účastníkmi a mnohé aj možnosť združovania ľudí s podobnými záujmami. Najpokrokovejšími typmi hier z tejto oblasti sú tzv. MMORPG, čo je skratka pre „Massively Multiplayer Online Role-Playing Game“. Tie združujú veľké množstvo užívateľov v obrovských virtuálnych svetoch, v ktorých platia určité zákony, funguje ekonomika a badať v nich mnoho analógií z reálneho sveta. Najznámejšou a najväčšou MMORPG hrou je podľa [3] World of Warcraft od firmy Blizzard s 8.5 miliónmi registrovaných hráčov po celom svete.

2.3.2 Vývoj sieťových počítačových hier

Na jednej strane ponúkajú sieťové počítačové hry bohatšie zážitky ako hry pre jedného hráča, no na strane druhej sú oveľa náročnejšie na vývoj. Stretneme sa tu s viacerými problémami, či už z oblasti návrhu, synchronizácie alebo férovosti. Je treba vyriešiť otázku komunikácie medzi jednotlivými

entitami aplikácie a to naviac s ohľadom na rýchlosť a plynulosť hry, ktorá je priamo závislá na kvalite sieťového pripojenia. Sieťové hry sú tvorené ako asynchrónne aplikácie, riadia sa teda zasielaním správ. Typickou architektúrou je klient-server architektúra, existuje však viacero spôsobov distribúcie informácii.

2.3.2.1 Všesmerové vysielanie

Jedným z nich je všesmerové vysielanie (broadcasting). Jedná sa o to, že každý počítač vysiela informácie o každej entite, ktorú spravuje, a to všetkým počítačom na sieti. Tieto správy sú prijaté všetkými účastníkmi hry a sú použité na aktualizáciu ich lokálnych kópií. Takto pracovali prvé sieťové hry, napr. v hre Doom vysielal pravidelne každý hráč stav svojej postavy. Podobne pracovalo veľa začínajúcich multi-player hier.

Tento spôsob pracuje v celku dobre pre malé siete, vo väčších už ale spôsobuje množstvo problémov. Najväčším problémom všesmerového vysielania je, že každý počítač v podsieti musí prijať a spracovať každý aktualizačný paket. Ak sa jedná o neverejné siete, nie je to až taký problém, ale vo väčších verejných sieťach, ktoré sa používajú aj na iné činnosti a priepustnosť siete je kritická, sa zbytočne zahlcujú nepotrebnými informáciami aj počítače, ktoré sa hry neúčastnia.

Ďalším problémom je, že vysielanie aktuálnej pozície, orientácie a prípadne ďalších stavových informácií, vyžaduje veľkú šírku pásma v prípade, že sa pozícia mení v každom kroku simulácie v hre. Akonáhle počet entít vysielajúcich takéto stavové informácie dosiahne určitý hraničný počet, systém prestáva pracovať, pretože šírka pásma jednoducho nebude stačiť.

2.3.2.2 Protokol DIS

Rozumnejšou variantou je použitie protokolu DIS (Distributed Interactive Simulation, distribuovaná interaktívna simulácia), ktorý minimalizuje vyťaženie sieťového spojenia zasielaním správ v minimalizovanej podobe, a to iba účastníkom hry, ktorí tieto správy potrebujú. Týmto spôsobom sa napr. neposiela neustále aktuálna pozícia entity pri každom simulačnom kroku hry, ale pošle sa iba údaj, pomocou ktorého sa aktuálna pozícia entity môže po určitú dobu vypočítavať, napr. vektor pohybu a aproximácia pohybu. Nemusia sa teda odosielať údaje v každom okamihu, stačí keď sa táto informácia pošle pri zmene vektora pohybu, napr. po užívateľskom zásahu. Takýto spôsob minimalizácie sieťového prenosu sa nazýva „dead reckoning“.

Každá entita rieši svoju úplnú simuláciu a tiež jednoduchý model „dead reckoning“. Uchováva si cestu k aktuálnej pozícii predvídanú dvoma modelmi. Pokiaľ sa tieto modely líšia o určitú hodnotu, je poslaná ďalšia správa o aktualizácii pre nastavenie každého modelu entity späť, do zhody s tým, čo entita práve vykonáva. Ďalej posiela každá entita periodicky aktualizačné správy, ktoré fungujú ako signál stálej aktivity (keep-alive). Táto správa má 3 účely. Pokiaľ sa niekto pripojí do simulácie po jej štarte, získa stavy všetkých entít v hre behom niekoľkých sekúnd. Pokiaľ sa aktualizačná správa z nejakého dôvodu stratí, systém sa obnoví bez problémov do niekoľkých sekúnd, pretože nová aktualizačná správa príde skoro, a tým aktualizuje stav danej entity. Pokiaľ nedochádzajú od nejakej

entity aktualizačné správy po dlhšiu dobu, môžu počítače v sieti predpokladať, že daná entita sa už nezúčastňuje na simulácii. Metóda „dead reckoning“ dosahuje vynikajúcich výsledkov, pretože aktualizácie sú posielané iba v prípade potreby. Množstvo prenosov a zahltenie šírky pásma sa teda minimalizuje. Systém je použitý aj v projekte TankGame, kde pracuje veľmi dobre, keďže bol navrhnutý na veľmi podobné účely (vojenské simulácie).

Problémom však nie je iba šírka pásma a latencia (odozva), ďalším obmedzujúcim faktorom je kompatibilita.

V DIS sa kompatibilita rieši pomocou definovania štandartu pre formát správ, ktoré sa používajú pre výmenu informácii medzi počítačmi zúčastňujúcimi sa simulácie. Tento štandart sa nazýva PDU (Protocol Data Unit). Existuje veľa rôznych typov PDU správ, od získania stavu jednotiek, po žiadosti o doplnenie munície.

2.3.3 Znaky sieťových počítačových hier

V sieťových počítačových hrách možno postrehnúť niekoľko typických znakov týkajúcich sa implementácie. Na niektoré vybrané z [4] sa pozrieme v tejto sekcii.

2.3.3.1 Tabuľka hráčov

V hre pre jedného hráča je stav každej entity a hráča kontrolovaný jedným počítačom. Programátor sa nemusí zaoberať konzistenciou, teda platnosťou stavov entít, pretože nehrozia žiadne nechcené zmeny týchto stavov. V sieťovej hre je však situácia iná. Stavy hráčov sa môžu meniť simultánne, teda naraz v jednom okamihu, na viacerých počítačoch. Aby sa zaistila konzistencia stavov entít v sieťovej hre, vytvorí sa tabuľka hráčov. V nej sú potom uložené informácie o každom hráčovi zúčastňujúcom sa na hre a je šírená na všetky zúčastnené počítače. Obsahuje tiež kontaktné údaje o hráčoch (IP adresa, port atd.), čo umožnuje prenos stavových informácií medzi centrálnym serverom a klientmi. Hlavná tabuľka je na serveri, ak niektorý z klientov urobí zmeny vo svojej kópii, pošlú sa tieto zmeny na server a odtiaľ sú distribované k ďalším klientom.

2.3.3.2 Sieťové udalosti

Sieťové hry, ako mnoho ďaľších asynchrónnych aplikácií, sú riadené udalosťami. To, ako sa hra chová, závisí na hráčových zásahoch. Udalosti teda vytvárajú interakciu. V hre pre jedného hráča je udalosť synonymom pre zmenu stavu určitej entity. Sieťová udalosť je rozšírenie tohto pojmu na sieťovú hru, kde zmena stavu entity musí byť distribuovaná ostatným hráčom na sieti. Sieťové udalosti sú prenášané po sieti pomocou kontaktných informácii uložených v tabuľke hráčov. Sieťová udalosť, ktorá mení stav hry na všetkých počítačoch, sa nazýva *zdieľaná udalosť*.

2.3.3.3 Správca sieťových udalostí

Správca sieťových udalostí je proces, ktorý čaká (načúva) na príchodzie sieťové udalosti a mení ich na udalosti lokálne, spravované lokálnym správcom udalostí na hráčovom počítači. Niekedy sa môže

stať, že v určitom okamihu príde viacero sieťoývch udalostí zároveň. Aby mohol správca sieťových udalostí tieto správy prijať a ďalej spracovať, musí byť konkurentný, musí teda vedieť obslúžiť viacero pripojení naraz. To sa docieli znásobením procesu správcu na viacero vlákien, čím sa zlepší efektívnosť obsluhy.

2.3.3.4 Zdieľaný objekt

Zdieľaný objekt je objektom hry, ktorého stav môže meniť a pozorovať viac ako jeden účastník, či už simultánne alebo v oddelených intervaloch. Príkladom zdieľaného objektu v hre TankGame je napr. ovládateľný tank. Jeho stav môže meniť hráč, ktorý ho ovláda, ale zároveň ho môžu ostatní hráči poškodzovať.

2.3.4 Spracovanie úloh v sieťových počítačových hrách

V prostredí sieťovej počítačovej hry je zadávanie a detekcia úloh skomplikovaná vďaka existencii viacerých hráčov v simulácii. Tí musia byť o zmenách stavu hry, a teda aj o plnení cieľov, pravidelne informovaní, aby sa zaistila konzistencia simulácie. Každá akcia na lokálnom počítači, ktorá zasahuje do zdieľaných objektov v hre, musí byť teda reflektovaná aj na ostatných počítačoch v sieti.

Splnenie zadaného cieľa jedným hráčom musí byť teda distribuované všetkým ostatným hráčom zúčastňujúcim sa simulácie. Predpokladajme, že každý hráč disponuje vlastným zoznamom úloh. Ak niektorý z hráčov splní úlohu, ktorá je spoločná pre viacerých užívateľov a môže byť splnená iba jedným z nich, vymaže sa z hráčového zoznamu úloh a ostatným je odoslaná správa, ktorou sa daná úloha v ich lokálnych zoznamoch zneplatní. Nebude ju teda možno splniť viacej krát. Zneplatnená úloha sa pri kontrolovaní zoznamu úloh ignoruje a považuje sa za nesplniteľnú.

Príklad je badateľný na nasledujúcej situácii. Na začiatku hry sa každému hráčovi zadá cieľ, ktorý pozostáva v obsadení vojenskej základne. V prostredí simulácie sa nachádza iba jedna základňa, jej stav môže pozorovať a meniť každý účastník, jedná sa teda o zdieľaný objekt. V okamžiku obsadenia základne jedným z hráčov sa táto informácia odošle všetkým ostatným účastníkom a následne je úloha o obsadení základne v ich lokálnych zoznamoch úloh zneplatnená. Zo zonamu sa však nemusí vymazávať, pretože sa napr. môže stať, že hráč, ktorý základňu obsadil, bude neskôr zničený, čím sa základňa stáva neobsadenou. Takáto situácia spôsobí vyslanie správy nielen o hráčovom zničení, ale aj o znovuobnovení platnosti úlohy týkajúcej sa obsadenia základne.

Ak dôjde k splneniu úlohy, ktorá je síce pre hráčov spoločná, ale je možno ju splniť viacerými hráčmi, vymazáva sa iba z ich lokálnych zoznamov úloh a správa o zneplatnení sa posielať nemusí. Ostatným hráčom v simulácii sa však musí poslať stavová aktualizácia hráča, ktorý úlohu splnil, aby boli informovaní o jej splnení. Na tento účel môže dobre poslúžiť tabuľka hráčov, ktorá udržuje informácie o každom hráčovi v hre. V nej sa aktualizuje položka o počte splnených úloh pre daného hráča, takže každý uvidí, že hráč splnil určenú úlohu.

Opäť poslúži na lepšie predstavenie si situácie jednoduchý príklad. Tentokrát je úlohou hráčov zničiť vojenskú základňu. Tá sa však po určitom čase opäť obnovuje, čo umožňuje splniť túto úlohu viacerým hráčom. Ak niektorý z hráčov základňu zničí, zapíše sa mu do tabuľky hráčov bod. Po tejto udalosti sa vyšle správa ostatným účastníkom a aktualizuje sa ich lokálna tabuľka hráčov, konkrétne položka uchovávajúca informácie o počte bodov daného hráča.

Ďalším typom cieľa môže byť úloha týkajúca sa iba vybraného hráča. Táto úloha nijako neovplyvňuje zdieľané objekty hry, takže sa ostatným hráčom nemusí ani posielať aktualizácia stavu daného hráča. Ostatným hráčom v hre sa táto úloha do zoznamu ani nepridá. Detekovanie splnenia cieľa potom prebieha klasicky, ako keby sa nejednalo o sieťovú hru.

Príkladom takejto úlohy môže byť úloha, ktorá sa splní, ak sa hráč dostaví na predom určené miesto. Týka sa však iba jeho a pre ostatných hráčov je „neviditeľná“ a irelevantná. Po jej splnení sa úloha vymaže z hráčovho lokálneho zoznamu bez odosielania akejkoľvek správy o tejto skutočnosti.

3 Systém zadávania a plnenia úloh v projekte TankGame

3.1 Projekt TankGame

Projekt TankGame je simuláciou boja tankov vo vygenerovanej virtuálnej krajine. Po implementácii sieťového kódu hry, sa stal projekt simuláciou distribuovanou, ktorej sa môžu účastniť viacerí užívatelia súčasne. Môžeme ho teda považovať za sieťovú počítačovú hru. TankGame je vyvíjaný profesormi a študentmi FIT VUT v Brně. Na tomto projekte pracujú študenti rôznych kurzov FIT VUT a študenti spracovávajúci bakalárske a diplomové práce na tejto fakulte.

3.2 Systém zadávania a plnenia úloh

Systém zadávania a plnenia úloh umožňuje zadávať viacerým hráčom rôzne úlohy a následne ich vyhodnocovať v zdieľanom virtuálnom prostredí hry TankGame. Je postavený na základe sieťovej časti hry TankGame vytvorenej Lukášom Obrdlíkom a popísanej v [1]. Je implementovaný pomocou niektorých typických vzorov popísaných v kapitole 2, ale vo veľkej časti ide vlastnou cestou. Projekt TankGame má už svoje základné mechanizmy vyriešené a cieľom tejto práce bolo do neho časť so systémom pre zadávanie a plnenie úloh iba pridať tak, aby zabehnuté a fungujúce časti projektu ostali nezmenené. Nie vždy sa to samozrejme podarilo dodržať, niekedy si situácia vyžadovala stávajúci kód mierne pozmeniť. Zmeny však boli prevedené tak, aby nijako neovplyvnili už fungujúce časti TankGame.

Systém umožňuje pracovať s ľubovoľnými úlohami a pre demonštráciu funkčnosti je naiplementovaný súboj viacerých hráčov po sieti, známy pod názvom „deathmatch“. Systém môže byť však použitý pre rôznorodé účely, môžu to byť jednak ďalšie multi-player módy hry alebo aj misie pre jedného hráča.

3.3 Základy systému

Základ systému tvorí trieda **`TMatch`.** V nej sa nachádza zoznam úloh **`ObjectiveList`,** implementovaný ako zoznam objektov typu **`TObjective`**. `TObjective` je bázovou triedou pre všetky typy úloh. Odvodením od tejto triedy môžu vznikať ľubovoľné úlohy. Každá úloha musí mať definovanú metódu **`CheckObjective`**, ktorá zisťuje, či bola úloha splnená. Pre bázovú triedu `TObjective` je táto metóda iba virtuálna a je na užívateľovi, aby si svojú vlastnú úlohu z tejto triedy odvodil a doprogramoval definíciu splnenia úlohy. Zadanie úloh prebieha počas zahájenia hry

funkciou **StartMatch**. Tu sa podľa typu hry rozhodne, aké a koľko úloh sa hráčovi zadá. Po naplnení zoznamu úloh pracuje systém nasledovne. Funkciou **CheckObjectiveList** sa v pravidelných intervaloch prechádza zoznam úloh pre konkrétneho hráča a pre každú úlohu sa volá metóda `CheckObjective`, aby sa zistilo, či je splnená. V okamžiku, keď je úloha splnená, vymazáva sa zo zoznamu úloh a prevádza sa predom určená reakcia. Ak sú takto odstránené všetky úlohy zo zoznamu, znamená to, že hráč splnil všetko, čo mu bolo zadané a tedal úspešne ukončil hru.

Pre lepšie vysvetlenie si ukážeme fungovanie systému pre prípad súboja viacerých hráčov pod názvom „deathmatch" v prostredí hry TankGame. V hre „deathmatch" je úlohou každého hráča *n*-krát zničiť protivníka skôr, ako túto úlohu splní niekto iný. Číslo *n* sa označuje ako „fraglimit", zničenie protihráča potom ako „frag". Po vytvorení hry sa každému účastníkovi hry vygeneruje zoznam úloh. Ten obsahuje úlohy zničenia tankov jednotlivých hráčov. Splnenie úlohy sa detekuje definovanou metódou `CheckObjective`, ktorá zisťuje, či bol nepriateľský tank, na ktorého bola úloha viazaná, zničený. Ak sa tak počas hry stalo, úloha sa zo zoznamu vymaže a zníži sa počítadlo zostávajúcich úloh. Hra končí v prípade, že niektorý z hráčov splnil *n* zadaných úloh.

3.4 Trieda TMatch

Trieda **TMatch** je základnym prvkom celého systému pre zadávanie a plnenie úloh. V aplikácii sa nachádza vždy najviac jeden objekt tejto triedy, jedná sa teda o tzv. „singleton". Trieda zapúzdruje zoznam úloh, metódy pre zadávanie a detekciu všetkých úloh, tabuľku hráčov, stavové informácie o hre a niektoré ďalšie metódy a atribúty potrebné pre fungovanie systému. Ukazateľ na objekt triedy `TMatch` je súčasťou hlavnej triedy celého projektu TankGame, triedy `TTankGame`, a má názov `Match`. Po vytvorení tohto objektu sa musí systém inicializovať a spustiť metódou `StartMatch`. Podľa jej parametrov sa dostáva simulácia do rôznych módov hry. V tejto metóde sa zadajú všetky potrebné úlohy pre úspešné dokončenie hry. Zadávanie prebieha vytváraním úloh odvodených od bázového typu `TObjective` a ich pridávaním do zoznamu úloh `ObjectiveList`. Tento zoznam sa potom prechádza v metóde `CheckObjectiveList` a kontrolujú sa jednotlivé splnené úlohy metódou `CheckObjective`. Volanie metódy `CheckObjectiveList` nastáva pri spustenej hre v každom simulačnom kroku TankGame, vo funkcii `TimeTick`. Indikátorom stavu spustenia hry je atribút triedy `TMatch Running`. Po splnení všetkých zadaných úloh v zozname `ObjectiveList` nastáva ukončenie hry metódou `EndMatch`. V stave ukončenej hry, ktorý indikuje atribút `Finished`, je potom ešte možné prezerať si výsledky hry vypisované na obrazovku pomocou objektu `MatchInformation`, ktorý je tiež atribútom triedy `TMatch`.

Na ukážku je vytvorený mód hry pre viacerých hráčov „deathmatch". Pri inicializácii hry sa metódou `StartMatch` vytvoria úlohy typu `TObjectiveDestroy` viazané na všetkých hráčov v simulácii a pridajú sa do zonamu úloh `ObjectiveList`. Ich počet závisí na hodnote atribútu

`Fraglimit`. `TObjectiveDestroy` je úloha odvodená od bázového typu úlohy `TObjective`. Jej cieľom jej zneškodnenie zadaného protivníka určeného premennou `target`. V každom kroku simulácie TankGame sa potom počas spustenej hry metódou `CheckObjectives` kontroluje zničenie tankov súperov. Po splnení určitého počtu úloh, daného atribútom `Fraglimit`, hra pre jedného hráča úspešne končí. Volá sa teda metóda `EndMatch`, ktorá nastaví premennú `Finished` do stavu logickej pravdy „true“.

3.4.1 Metódy triedy TMatch

3.4.1.1 TMatch( )

Konštruktor triedy `TMatch` slúži na predinicializáciu hry, či už sa jedná o hru pre jedného alebo viacerých hráčov po sieti. Nastavuje do počiatočného stavu premennú `Running`, ktorá indikuje spustenie hry. Priraďuje jej hodnotu „false“. Podobne priraďuje hodnotu logickej nepravdy aj premennej `Finished`, ktorá indikuje stav ukončenia hry. Ďalej inicializuje premennú `ReadyCount`, ktorá udržuje informáciu o počte pripravených hráčov pri spúšťaní hry. Jej presnejšia funkcia bude demonštrovaná neskôr. Do tejto premennej je opäť priradená logická hodnota „false“. Nakoniec nasleduje inicializácia ukazateľa na objekt typu `TFont2D` s názvom `MatchInformation`. Tento ukazateľ slúži na výpis informácii o stave hry na obrazovku. Pre hru „deathmatch“ nastavuje počiatočnú hodnotu atribútu `Fraglimit`.

Konštruktor triedy `TMatch` sa volá pri vytváraní objektu `Match`, ktoré nastáva po výbere položky „Start Match“ z menu „Multiplayer“.

Ak sa jedná o mód hry pre viacerých hráčov, ako napr. v prípade hry „deathmatch“, musí byť pred vytvorením tohto objektu zaslaná správa o započatí hry všetkým účastníkom simulácie. V tejto správe dostávajú zároveň informácie potrebné k započatiu hry. Po obdržaní správy sa teda u každého hráča vytvorí objekt `Match` a pomocou doplnkových informácii o type hry, pravidlách a pod., sa hra spustí.

3.4.1.2 ~TMatch( )

Deštruktor triedy sa stará o vymazanie dynamicky alokovaných objektov, ktoré počas vytvorenia objektu a jeho činnosti v aplikácii vznikli. Jedná sa o záznamy v tabuľke hráčov `PlayerTable`, ďalej objekt slúžiaci na výpis informácii o stave hry na obrazovku `MatchInformation`, a nakoniec je potrebné vymazať všetky zadané a nesplnené úlohy, ktoré sa počas hry neodstránili zo zonamu. To má na starosti metóda `DeleteObjectives`. Tieto objekty je potrebné pri ukončení hry vymazať, pretože by inak ostávali vytvorené v pamäti aj po ukončení hry a nebolo by už možné sa na ne odkázať nijakým ukazateľom. Zbytočne by tak zaberali pamäťové miesto, ktoré by mohla

simulácia TankGame ďalej využívať. Jednalo by sa o „memory leak“, ktorý je v každej aplikácii nežiadúcim.

Deštruktor sa volá pri mazaní objektu triedy `TMatch` z pamäti, teda pri volaní `delete` na objekt `Match`. To nastáva pri vybratí ponuky „End Match“ z menu "Multiplayer" a opustení simulácie položkou menu „Exit“. Pred samotným vymazaním objektu z pamäti sa zvyčajne volá metóda `EndMatch` a posiela sa správa o ukončení hry.

Rovnako ako pri vytváraní objektu `Match`, aj tu sa musí v móde pre viacerých hráčov odoslať správa o ukončení hry, ktorá zaručí vymazanie objektu aj u ostatných hráčov.

3.4.1.3 CheckObjectiveList

Táto metóda sa stará o detekciu cieľov zadaných v zozname úloh `ObjectiveList`. Jej návratovou hodnotou je počet nesplnených úloh v zozname. Metóda prechádza `ObjectiveList` záznam po zázname a s využitím polymorfizmu volá, cez objekt bázovej triedy `TObjective`, virtuálnu funkciu na detekciu splnenia cieľa `CheckObjective`. Týmto spôsobom sa zavolá konkrétna funkcia `CheckObjective` pre každú úlohu aj keď sa v zozname nachádzajú úlohy rôznych typov.

Po detekovaní splnenia úlohy sa potom prevedie akcia závisiaca na konkrétnom móde hry. V prípade módu hry pre viacerých hráčov „deathmatch“ sa najprv odošle správa o hráčovom splnení úlohy. Po obdržaní tejto správy si ostatní účastníci simulácie aktualizujú svoje tabuľky hráčov. Ďalšou reakciou na splnenie úlohy je oznam hráčovi, ktorý úlohu splnil, v podobe výpisu na obrazovku. Taktiež jeho tabuľka hráčov sa musí aktualizovať.

Splnená úloha sa najprv odstráni zo zoznamu úloh, aby nebola ďalej kontrolovaná. V prípade zdieľanej úlohy, ktorá sa týka viacerých hráčov a je možné ju splniť iba jedným z hráčov, je treba odoslať ostatným účastníkom správu o jej zneplatnení. Po odstránení úlohy zo zoznamu je objekt úlohy vymazaný z pamäti, pretože pre hru už ďalej nebude potrebný.

V niektorých prípadoch je potrebné niektoré úlohy v zozname zneplatniť. Konkrétne v prípade hry „deathmatch“ musí hra zneplatniť úlohy týkajúce sa určitého hráča po zničení jeho tanku, a to do až doby, kým si hráč nevytvorí nový tank. Zneplatnenie úloh sa prevádza metódou úlohy `MakeInvalid`.

Po nájdení splnenej úlohy v zozname sa jeho prehľadávanie ukončuje a metóda vracia počet nesplnených úloh v zozname.

Metóda sa volá pravidelne v každom kroku simulácie TankGame, vo funkcii `TimeTick`.

3.4.1.4 StartMatch

Metóda `StartMatch` inicializuje a spúša hru resp. niektorý z módov hry, ktorý je určený parametrom `gametype`. Prvými krokmi metódy sú inicializácie stavových premenných v objekte `Match`. Inicializuje sa ukazateľ na zoznam hráčov v simulácii `PlayerList`, typ hry `Gametype`, vytvorí sa tabuľka hráčov `PlayerTable` a naplní sa implicitnými záznamami jednotlivých hráčov.

Potom sa podľa módu hry naplní zoznam úloh pre daného hráča. Počet úloh, ich typ a ostatné pravidlá závisia od implementácie daného módu. V prípade hry „deathmatch“ sa najprv nastaví „fraglimit“ (reprezentovaný atribútom `Fraglimit`), tzn. počet zneškodnení súperových tankov, po ktorých hráč vyhráva hru. Podľa tohto počtu sa priradia do zoznamu úlohy typu `TObjectiveDestroy`, ktorých cieľom je práve zneškodnenie súperových tankov. Hra môže skončiť niekoľkonásobným zničením jedného súpera, alebo zničením rôznych súperov. Preto treba zadať toľko úloh na zničenie každého súpera, koľko je číslo „fraglimit“. Prejde sa teda zoznam všetkých hráčov v hre a do zoznamu úloh sa hráčovi pridá daný počet úloh na zničenie každého súpera.

Po úspešnom zadaní všetkých potrebných úloh sa nastavuje návratová hodnota `Running` na logickú hodnotu „true“ a metóda vrátením tejto stavovej informácie končí.

Ak bol touto metódou spustený mód hry pre jedného hráča a metóda `StartMatch` vrátila „true“, hra je úspešne spustená, detekcia cieľov v tejto chvíli už beží a hráč môže pokojne spĺňať zadané úlohy.

V prípade módu hry pre viacerých hráčov je však situácia komplikovanejšia. Informácia o spustení hry sa najprv musí rozšíriť medzi ostatných účastníkov simulácie. Vyšle sa teda správa typu `TMatchCreatePDU`, ktorá indikuje vytvorenie a spustenie hry a obsahuje aj údaje o type hry. Po obdržaní tejto správy ostatnými účastníkmi, si aj oni vytvoria objekt triedy `TMatch` a pomocou informácii o type hry spustia daný mód hry metódou `StartMatch`.

3.4.1.5 EndMatch

Metóda `EndMatch` ukončuje aktuálne spustený mód hry nastavením stavovej premennej `Running` do stavu logickej hodnoty „false“. Tým sa preruší detekcia cieľov v každom kroku simulácie a vytvorená hra sa považuje za ukončenú. V prípade módu hry pre viacerých hráčov sa ale musia aj ostatní účastníci simulácie dozvedieť o ukončení hry. Preto sa im po samotnom ukončení odosiela správa o tomto stave. Po jej prijatí aj oni ukončujú hru.

Objekt `Match` však ešte naďalej ostáva vytvorený v pamäti a na obrazovke sa stále zobrazujú informácie o skončenej hre. V tomto čase je hra ukončená, detekcia cieľov neprebieha, nemožno vytvárať nové tanky, hráči iba majú možnosť prehliadnuť si výsledky hry. Pre úplné vypnutie hry treba z pamäte vymazať objekt `Match`. To sa deje napr. pri vybratí ponuky „End Match“ z menu "Multiplayer". O tomto kroku však treba, rovnako ako v prípade ukončenia hry, informovať ostatných zúčastnených, aby objekt `Match` vymazali, ak sa jedná o mód hry pre viacerých hráčov.

Metóda zároveň nastavuje aj stavovú premennú `Finished` na logickú hodnotu „true“. Táto premenná má svoju funkciu v niektorých vedľajších mechanizmoch, napr. pri vytváraní nového tanku, kde zabraňuje vytvoreniu ďalšieho tanku po skončení hry.

Metóda sa volá v prípade, že v zozname úloh už nezostávajú žiadne splniteľné úlohy, alebo ak hráč splnil presne taký počet úloh, aký je potrebný na ukončenie hry. Napr. pri hre pre viacerých

hráčov „deathmatch“ bude hra ukončená v okamihu, keď niektorý zo zúčastnených hráčov splní počet úloh rovnajúci sa číslu „fraglimit“.

3.4.1.6 RenderInfo

Metóda `RenderInfo` sa stará o výpis informácii o aktuálnom móde hry na obrazovku. Využíva pri tom objekt `MatchInformation` typu `TFont2D`, ktorý bol už v TankGame v minulosti implementovaný. Na obrazovku sú vypisované základné údaje o hre, v prípade hry „deathmatch“ je to „fraglimit“ a údaje z tabuľky hráčov o počte splnených úloh jednotlivých hráčov. Stavové informácie sú na obrazovku vypisované metódou `DrawText`, ktorá je súčasťou triedy `TFont2D`.

Táto metóda sa volá pravidelne pri každom kroku simulácie TankGame vo funkcii `TimeTick`.

3.4.1.7 GetCompletedObjectives

Táto metóda vracia počet úspešne splnených cieľov zo zonamu úloh. Tento počet sa získava z tabuľky hráčov. Keďže pri vytváraní tabuľky hráčov je vždy prvé miesto vyhradené hráčovi, u ktorého je tabuľka vytvorená, vyberie sa prvý hráč z tejto tabuľky a vráti sa jeho záznam o počte splnených úloh. Pri hre „deathmatch“ je to počet zneškodnení nepriateľských tankov.

Metóda sa môže využívať pri kontrole úspešnosti daného hráča v prípade, že údaje o počte splnených úloh nie sú zaznamenávané do tabuľky hráčov.

3.4.1.8 DeleteObjectives

Môže sa stať, že je hra ukončená ešte predtým, ako hráč splnil zadané úlohy. Táto situácia nastáva napr. pri výbere ponuky „End Match“ z menu "Multiplayer", alebo pri úspešnom ukončení hry iným hráčom. Nesplnené úlohy potom ostávajú vytvorené v pamäti v zozname úloh. Aby nezaberali zbytočne po skončení hry miesto v pamäti, vymazávajú sa metódou `DeleteObjectives`.

Táto metóda sa volá v deštruktore triedy `TMacth`, teda pri vymazávaní objektu `Match` z pamäti.

3.4.1.9 IsRunning

Metóda `IsRunning` vracia stav vytvorenej hry, ktorý je uložený v premennej `Running`. Návratovou hodnotou je logická pravda „true“ v prípade spustenej hry. Ak hra ešte nebola spustená funkciou `StartMatch`, alebo je už hra ukončená, vracia metóda hodnotu logickej nepravdy „false“.

Metóda sa využíva pri kontrolovaní stavu spustenia hry pri výpise stavových informácii hry na obrazovku, ďalej riadi detekovanie cieľov v každom kroku simulácie, zohráva svoju úlohu aj pri vytváraní nových tankov a všeobecne všade tam, kde je potrebné zistiť, či je spustená hra.

3.4.1.10 IsFinished

Metóda `IsFinished` má podobnú úlohu ako metóda `IsRunning`, s tým rozdielom, že vracia stav hry uložený v premennej `Finished`, reprezentujúci ukončenie hry. Ak je hra ukončená, ale objekt hry ešte nie je vymazaný z pamäti, vracia logickú hodnotu „true“. Takáto situácia nastáva po splnení všetkých zadaných cieľov hry, kedy je ďalší čas v hre určený už len na prezeranie výsledkov. Ak hra ešte nie je ukončená, metóda vracia logickú hodnotu „false“.

3.4.1.11 GetPlayerList

Metóda vracia ukazateľ na zoznam hráčov v hre. Môže byť použitá na kontrolu počtu hráčov v hre, pretože niektoré z módov hry môžu svojimi pravidlami tento počet obmedzovať. Návratovou hodnotou je ukazateľ na celý zoznam hráčov typu `TPlayer` pripojených do simulácie TankGame. Ukazateľ je uložený v atribúte `PlayerList`.

3.4.2 Atribúty triedy TMatch

3.4.2.1 Tabuľka hráčov PlayerTable

Trieda `TMatch` obsahuje tabuľku hráčov **`PlayerTable`** pozostávajúcu zo zoznamu hráčov a počte ich splnených úloh. Tu budú v prípade iného módu hry ďalšie informácie potrebné pre aktualizáciu ostatných hráčov. Údaje v tabuľke hráčov sa menia v priebehu hry a každá zmena v nej prevedená musí byť ohlásená aj ostatným hráčom. Ak jeden hráč prevedie v tabuľke zmenu musí sa táto zmena odoslať všetkým zúčastneným, aby mohli svoje lokálne tabuľky hráčov aktualizovať. V prípade módu hry „deathmatch“ musí hráč pri každom zneškodnení nepriateľského tanku odoslať o tejto skutočnosti správu všetkým ostatným hráčom. Tí si potom pri jeho zázname v tabuľke zvíšia položku o počte hráčových splnených úlohách o 1.

Keďže TankGame už jednu podobnú tabuľku hráčov obsahuje, nemusia byť v `PlayerTable` uchovávané kontaktné informácie o hráčoch. Stačí uchovávať tie stavové údaje, ktoré sú dôležité pre aktuálny mód hry. Podľa teórie by mala existovať jedna centrálna tabuľka na serveri a tabuľky u jednotlivých hráčov by mali byť lokálne Pri zmene v lokálnej tabuľke by sa táto malo preniesť na server a odtiaľ k jednotlivým hráčom na sieti. V našom prípade sú všetky lokálne tabuľky na rovnakej úrovni, nie je potreba vytvárať centrálnu tabuľku a tú potom šíriť medzi ostatnými počítačmi. Pri zmene niektorej lokálnej tabuľky sa všetkým účastníkom simulácie pošle správa a tí si tabuľku aktualizujú. Toto riešenie je jednoduchšie a menej náročné na sieťové prenosy, ako keby sa mala prenášať celá tabuľka. V niektorých prípadoch, napr. pri veľkom počte zmien, môže byť ale výhodnejšie prenášať celý záznam o konkrétnom hráčovi alebo dokonca aj celú tabuľku.

Záznamy v tabuľke hráčov sú typu **`TPlayerRecord`**. Jedná sa o štruktúru obsahujúcu meno hráča, jeho identifikačné číslo v simulácii a údaj o počte splnených úloh, alebo všeobecne ľubovoľnú

informáciu potrebnú pre daný typ hry. Záznamy sú usporiadané v zozname typu `TArrayList`, ktorý bol v TankGame už vytvorený.

3.4.2.2 Zoznam úloh ObjectiveList

Zoznam úloh `ObjectiveList` je typu `TArrayList<TObjective>`. Jedná sa teda o zoznam objektov úloh bázového typu `TObjective`. V zozname sa samozrejme môžu nachádzať aj úlohy iného typu, ich trieda však musí byť odvodená od tohto základného typu. `ObjectiveList` reprezentuje zoznam zadaných cieľov pre daného hráča, u ktorého je objekt `Match` vytvorený. Sem sa pri spustení hry uložia všetky úlohy, ktorých splnenie predstavuje pre hráča úspešné dokončenie hry. Tento atribút sa využíva v metóde triedy `TMatch` `CheckObjectives`, pomocou ktorej sa pri detekcii splnených cieľov v každom kroku simulácie TankGame tento zoznam prechádza a na jednotlivé úlohy sa volá metóda kontrolujúca jej splnenie.

Pri mazaní objektu triedy `TMatch` z pamäti, presnejšie povedané pri volaní deštruktoru triedy `TMatch`, sa tento zoznam vyprázdňuje, vymazávajú sa z neho nesplnené úlohy a následne zaniká. Vyprázdňovanie sa konkrétne deje v metóde `DeleteObjectives`.

Zadávanie nových úloh prebieha pridaním nového objektu úlohy do zoznamu. Keďže je zoznam typu `TArrayList`, prebieha toto metódou tohto typu, a to konkrétne metódou `Add`. Príklad zadania novej úlohy v hre „deathmatch“, pozostávajúcej zo zneškodnenia protivníkového tanku, vyzerá nasledujúco:

```
...
TObjective* DestroyPlayer = new TObjectiveDestroy(player);
ObjectiveList.Add(DestroyPlayer);
...
```

Prvý riadok ukazuje vytvorenie novej úlohy odvodeného typu `TObjectiveDestroy` a riadok druhý samotné zadanie úlohy pridaním do zoznamu `ObjectiveList`.

Odstránenie úlohy zo zoznamu, či už po jej úspešnom splnení alebo v inom prípade, prebieha tiež metódami typu `TArrayList`. Na odstránenie prvku zo zoznamu sa použije metóda `Remove`. Príklad na hre „deathmatch“ opäť demonštruje jej použitie.

```
...
ObjectiveList.Remove(objective);
delete objective;
...
```

Na prvom riadku sa úloha odstráni zo zoznamu, na druhom vymaže z pamäti.

3.4.2.3 Zoznam hráčov PlayerList

Okrem tabuľky hráčov obsahuje trieda `TMatch` aj zoznam hráčov `PlayerList`. Ten pozostáva zo záznamov typu `TPlayer`, čo je štruktúra definovaná triedou `TTankGame`. Tento typ je už

v TankGame zaužívaný a hlavná trieda `TTankGame` ho používa na implementáciu svojho zoznamu hráčov nachádzajúcich sa v simulácii. V triede `TMatch` sa ukladá iba ukazateľ na tento zoznam. Nastavuje sa pri volaní metódy `StartMatch` a ako návratová hodnota je vrátený pri volaní metódy `GetPlayerList`.

Využíva sa pri vytváraní tabuľky hráčov, pri zadávaní úloh do zoznamu `ObjectiveList`, detekcii úloh a pri niektorých ďalších úkonoch.

3.4.2.4 Typ hry Gametype

Atribút `Gametype` udržuje informáciu o móde aktuálne spustenej hry. Záznam je typu `TGametype`, čo je vymenovaný typ, zatiaľ implementovaný iba s hotnotami `gtDeathMatch` a `gtOther`. Hodnota `gtDeathMatch` predstavuje mód hry pre viacerých hráčov „deathmatch“, hodnota `gtOther` je v type iba na ukážku a môže predstavovať ľubovoľný iný typ hry. Typ `TGametype` je však otvorený pre ďalšiu implementáciu. So vznikom nových módov hry, tu budú pribúdať nové hodnoty.

Hodnota atribútu `Gametype` sa nastavuje pri spúšťaní hry v metóde `StartMatch`. Podľa tejto premennej sa riadi metóda `StartMatch` pri zadávaní úloh a taktiež metóda `CheckObjectiveList` pri detekovaní cieľov.

3.4.2.5 Stavová premenná Running

`Running` je atribútom triedy `TMatch` typu `bool`, ktorý ukladá informáciu o stave hry, konkrétne údaj o tom, či je hra úspešne spustená. Jeho počiatočná hodnota nastavovaná konštruktorom triedy `TMatch` je hodnota logickej nepravdy „false“. Tento atribút sa nastavuje pri spúšťaní hry metódou `StartMatch`. Má hodnotu logickej pravdy „true“ ak je hra úspešne spustená, a naopak, pri ešte nespustenej alebo už ukončenej hre, má hodnotu logickej nepravdy „false“. Po úspešnom zadaní všetkých úloh v metóde `StartMatch` sa nastavuje na hodnotu „true“, ak sa nepodarí zadať všetky úlohy, tak ako bolo plánované, nastaví sa na hodnotu „false“. Atribút je zároveň návratovou hodnotou metódy `StartMatch` a `IsRunning`.

Atribút riadi detekciu splnených úloh. Ak je nastavený na hodnotu „true“, prebieha detekovanie cieľov v metóde `TimeTick` hlavnej triedy simulácie TankGame. V prípade hodnoty „false“ v premennej, kontrola splnených úloh neprebieha.

Pri ukončovaní hry metódou `EndMatch` sa táto premenná nastavuje na hodnotu logickej nepravdy „false“.

3.4.2.6 Stavová premenná Finished

Podobne, ako atribút `Running`, aj atribút `Finished` predstavuje určitý stav hry, presnejšie sa jedná o stav ukončenia hry. Atribút je taktiež booleanovského typu `bool`, môže teda nadobúdať hodnoty

logickej pravdy „true“ alebo logickej nepravdy „false“. Jeho počiatočná hodnota je „false“, pretože pri volaní konštruktoru triedy `TMatch` sa iba vytvorí objekt tejto triedy. Hodnotu „true“ nadobúda atribút až pri ukončení hry metódou `EndMatch`.

Atribút sa využíva najmä pri výpise stavových informácii o hre po jej skončení. Uplatnenie však má aj pri kontrole vytvárania nových tankov po skončení hry, kde zabraňuje vytváraniu ďalších jednotiek. Premenná `Finished` je návratovou hodnotou metódy `IsFinished`.

3.4.2.7 Výpis stavu hry na obrazovku z MatchInformation

Atribút `MatchInformation` je objektom typu `TFont2D`, ktorý slúži na výpis stavových informácii o hre na obrazovku. Tento výpis prebieha ihneď po spustení hry metódou `StartMatch` a pretrváva aj po ukončení hry metódou `EndMatch`. Využíva sa v metóde `RenderInfo`, ktorá má na starosti zhromaždenie potrebných stavových informácii o hre a na nej zúčastňujúcich sa hráčoch, a následný výpis údajov na obrazovku. To prebieha práve cez objekt `MatchInformation` metódou `DrawText`.

3.5 Trieda TObjective

Trieda **`TObjective`** predstavuje všeobecný typ úlohy, ktorá je pre hráča v hre splniteľná. Je iba vzorovou triedou ľubovoľnej splniteľnej úlohy pre konkrétne, užívateľsky nadefinované typy úlohy. Tieto nové typy úloh je potreba od bázovej triedy `TObjective` odvodiť. Trieda obsahuje virtuálnu metódu **`CheckObjective`**, ktorá má na starosti detekciu splnenia úlohy. Tú je treba v odvodenom type úlohy predefinovať tak, aby zaisťovala kontrolu splnenia úlohy pre daný typ úlohy.

Úloha môže byť viazaná na niektorého z hráčov v simulácii. Toto využíva na ukážku odvodená úloha `TObjectiveDestroy`. Naviazanie na hráča zabezpečuje atribút `target` typu `TPlayer`. Táto úloha sa využíva v hre „deathmatch“, kde je týmto atribútom určený hráč, ktorého zneškodnenie tanku predstavuje cieľ úlohy. Na zistenie hráča, na ktorého je úloha viazaná, sa používa metóda `GetTargetPlayer`, ktorá vracia ukazateľ na hráča typu `TPlayer`. Ak je potrebné naviazať úlohu na ľubovoľný objekt, stačí odvodiť nový typ úlohy z bázového typu `TObjective` a do nového typu pridať atribút zabezpečujúci naviazanie na daný objekt.

Nová úloha sa vytvára konštruktorom triedy odvodenej od základnej triedy `TObjective`. V prípade odvodenej úlohy `TObjectiveDestroy` sa ako parameter konštruktoru určí hráč, na ktorého je úloha viazaná. Naviazanie úlohy na hráča sa môže previesť aj neskôr, a to metódou `SetTargetPlayer`. Zadanie úlohy vybranému hráčovi potom prebieha zaradením úlohy do jeho zoznamu úloh `ObjectiveList`. Kontrola stavu a splnenia tejto úlohy prebieha v metóde `CheckObjectiveList`.

V priebehu hry sa môže stať, že vybraná úloha, zadaná hráčovi na začiatku hry ako splniteľná, sa počas prebiehajúcej hry stane pre daného hráča naďalej nesplniteľnou. Táto situácia môže nastať, ak sa jedná o úlohu zdieľanú, teda splniteľnú pre všetkých hráčov v hre, a zároveň o takú úlohu, ktorá môže byť splnená iba jediný krát. Keďže je zadaná v rovnakej podobe všetkým hráčom na začiatku hry, je potreba ju po jej splnení niektorým z hráčov zneplatniť v zozname úloh ostatných hráčov. Pre túto príležitosť obsahuje bázová trieda `TObjective` atribút **`valid`**. Zapísaním logickej hodnoty „false“ do tohto atribútu alebo použitím metódy `MakeInvalid` sa stáva úloha neplatnou. Neplatná úloha je pri detekcii splnenia úloh v zozname úloh ignorovaná a považuje sa za nesplniteľnú. Ak potrebujeme po čase úlohu znovu preniesť do stavu platného, nastavíme atribút `valid` na hodnotu logickej pravdy „true“ alebo použijeme metódu `MakeValid`. Na zistenie platnosti úlohy slúži metóda `IsValid`, ktorá vracia „true“, ak je úloha platná, v opačnom prípade vracia metóda hodnotu logickej nepravdy „false“.

3.6 Komunikácia v systéme pre zadávanie a plnenie úloh pri hre pre viacerých hráčov

Pri módoch hry pre viacerých hráčov je v systéme pre zadávanie a plnenie úloh dôležitým faktorom komunikácia. Keďže simulácia projektu TankGame je simuláciou distribuovanou, prebieha teda na viacerých počítačoch a môžu sa do nej zapojiť viacerí užívatelia. Zdieľanie jedného virtuálneho priestoru simulácie so sebou prináša nutnosť zaistiť konzistenciu simulácie na všetkých počítačoch v sieti. Stav simulácie teda musí byť na všetkých počítačoch v každom kroku simulácie rovnaký, resp. stav simulácie na jednom počítači musí zodpovedať stavu simulácie na počítači druhom. Úplná konzistencia sa zaisťuje veľmi ťažko. Často je lepšie zvoliť nepresnejšie riešenie, ktoré je ale na druhej strane menej náročnejšie na prenosy na sieti. Tento postup zvolil aj tvorca sieťového kódu v projekte TankGame [1] a využil princípy protokolu DIS, ktorý je popísany v kapitole 2.3.

3.6.1 Komunikácia pomocou správ

Entity v distribuovanej simulácii TankGame medzi sebou komunikujú zasielaním správ typu PDU (Protocol Data Unit). Tieto správy tvoria akési rozhranie medzi objektmi, ktoré sú spravované priamo na počítači, kde vznikli, a ich kópiami u ďalších účastníkov simulácie na ostatných počítačoch v sieti.

Každá takáto správa obsahuje informácie o entite, ktorej sa týka. V jej tele je tiež určené, do ktorej z viacerých implementovaných množín správa patrí. Množina správ sa skladá z troch podmnožín. Jednou z nich tvoria správy pre vytvorenie simulačnej entity. Ďalšou skupinou sú správy pre aktualizáciu stavu vybranej simulačnej entity. Poslednou podmnožinou sú správy využívajúce sa pri vzniknutí určitej udalosti na entite. Tento typ správ sa bude v systéme pre zadávanie a detekciu úloh vyskytovať najčastejšie.

3.6.2 Správy použité v systéme pre zadávanie a detekciu úloh

Systém využíva niekoľko druhov správ na komunikáciu medzi entitami pri vytváraní hry a pri samotnej hre. Reakcie na tieto správy sa nachádzajú v metóde `DispatchMessage` hlavnej triedy simulácie `TTankGame`.

3.6.2.1 TMatchCreatePDU

Prvým typom správy, ktorý systém využíva, je `TMatchCreatePDU`. Tento typ bol odvodený od triedy `TCreatePDU` a predstavuje správu o vytvorení hry. Správa sa vysiela ostatným účastníkom simulácie po úspešnom vytvorení a spustení hry. Po prijatí tejto správy si klient, ktorý správu obdrží, vytvorí objekt `Match` a spustí hru metódou `StartMatch` pomocou informácii obsiahnutých v správe. Správa obsahuje informácie o type hry a prípadne ďalšie údaje potrebné pre konkrétny mód hry ako napr. pravidlá a pod. V prípade hry „deathmatch“ je v správe obsiahnutá informácia o type hry „deathmatch“ a údaj o hodnote „fraglimitu“. Vytvorenie hry sa prevádza vo funkcii `CreateMatch`, ktorej parametrom je správa typu `TMatchCreatePDU`.

3.6.2.2 Správa udalosti etMatchStart

Táto správa je veľmi podobná predchádzajúcej správe `TMatchCreatePDU`. Vyjadruje udalosť založenia hry a využíva sa pri vytváraní hry po výbere ponuky „Start Match“ z menu "Multiplayer".
Na túto správu však reaguje iba server, teda ten počítač, ktorý hru zakladá. Správu posielajú serveru klienti, ktorí takto potvrdzujú svoju pripravenosť na začatie hry. Po prijatí správ od všetkých klientov v simulácii, je všetko pripravené, tanky jednotlivých hráčov sú rozložené a server môže poslať všetkým klientom správu typu `TMatchCreatePDU` a spustiť hru. Údaj o počte pripravených hráčov je uchovávaný v atribúte `ReadyCount`.

Správa je typu `TEventPDU`, čo predstavuje správu udalosti v simulácii. Ďalšie popisované správy budú taktiež tohto typu.

3.6.2.3 Správa udalosti etMatchEnd

Správa udalosti `etMatchEnd` sa posiela v prípade ukončenia hry. Deje sa tak pri úspešnom dokončení hry jedným z hráčov, alebo pri nútenom ukončení, ktoré nastáva po výbere ponuky „End Match“ z menu „Multiplayer“. Po prijatí tejto správy ukončuje klient aktuálne vytvorenú hru a podľa hodnoty atribútu `value` v správe rozhoduje o vyhodnotení výsledkov hry. Ak je v premennej `value` hodnota „0“, stav hry sa nevyhodnocuje. Deje sa tak pri nútenom ukončení hry, kedy nemá zmysel určovať víťaza hry. Ak je ale v premennej `value` hodnota „1“, stav hry sa vyhodnotí.

Táto správa hru iba ukončuje, pre úplné skončenie hry a vymazanie objektu `Match` je potreba poslať ešte správu `etMatchDelete`.

3.6.2.4 Správa udalosti etMatchDelete

Odoslanie správy udalosti `etMatchDelete` oznamuje klientovi, ktorý správu prijme, že je aktuálna hra ukončená a pripravená na úplné skončenie a vymazanie. Jeho reakciou je teda vymazanie objektu `Match` z pamäti a nastavenie ukazateľa na hodnotu `NULL`.

Táto správa sa posiela po výbere ponuky „End Match“ z menu „Multiplayer“.

3.6.2.5 Správa udalosti etDeleteTank

`etDeleteTank` je správou udalosti, ktorá vyjadruje žiadosť o vymazanie tanku zo simulácie. Objekt tanku je určený jednoznačným a unikátnym číslom ID, ktoré sa získa metódou `GetID` triedy `TUnit`. Toto číslo sa priradí do atribútu správy `objectID`, ktorý je súčasťou štruktúry `entityRecord`. Zaslanie správy sa využíva pri vytváraní nového tanku do hry, kde je aktuálne vytvorený tank hráča zničený. Tento tank treba pred vytvorením nového najprv vymazať zo simulácie. Po prijatí tejto správy odstráni klient určený tank zo zoznamu jednotiek `UnitList` a následne ho vymaže z pamäti. Zároveň vyšle správu `etTankDeleted`, ktorá potvrdzuje vymazanie tanku.

Pri vytváraní novej hry výberom ponuky „Start Match“ z menu „Multiplayer“ sú všetky jednotky na scéne vymazané a vytvára sa jeden tank pre každého hráča. Vymazanie tankov však musí prebiehať pomocou správ `etDeleteTank` a `etTankDeleted`, inak by nebola zaručená konzistencia simulácie. Pri spúšťaní hry je potreba zaoberať sa aj ukazateľom `ActiveUnit` hlavnej triedy `TTankGame`, a pri vymazávaní tanku, ktorý je označený ako aktívna jednotka (ukazuje na neho ukazateľ `ActiveUnit`) je treba túto jednotku deaktivovať metódou `Deactivate` a nastaviť ukazateľ na `NULL`. O tom, či sa bude deaktivovanie aktívnej jednotky pri prijatí správy brať do úvahy, rozhoduje atribút `value` v štruktúre správy `TEventPDU`. Ak je jeho hodnota „1“, aktívna jednotka je deaktivovaná. Pri hodnote value „0“ sa deaktivovanie jednotky neprevádza.

3.6.2.6 Správa udalosti etTankDeleted

Správa udalosti `etTankDeleted` je potvrdzujúcou správou o vymazaní tanku. Posiela sa vždy pri reakcii na správu `etDeleteTank`, a to po odstránení tanku zo zoznamu jednotiek `UnitList` a vymazaní tanku z pamäti. Správa indikuje, že inštancia tanku určeného hodnotou atribútu `objectID` štruktúry `entityRecord`, je na jednej klientskej strane vymazaná a môže sa vymazať aj na strane druhej. Prejde sa tede zoznam jednotiek `UnitList` a podľa ID jednotky sa nájde tank určený na vymazanie. Následne sa odstráni zo zoznamu a vymaže z pamäti.

Po reakcii na túto správu je inštancia tanku určená správou `etDeleteTank` odstránená zo simulácie, jednak na strane odosielateľa, a taktiež sú odstránené kópie tohto tanku na ostatných počítačoch.

3.6.2.7 Správa udalosti etDeployTank

Správa udalosti `etDeployTank` vyjadruje žiadosť o vytvorenie nového tanku. Klient si po prijatí tejto správy vytvorí nový tank a zašle správu o jeho vytvorení typu `TTankCreatePDU`, aby si mohli ostatní účastníci simulácie vytvoriť jeho kópiu a pridať si ho do svojich lokálnych zoznamov jednotiek. Po odoslaní tejto správy sa ešte prevedú všetky potrebné operácie na zfunkčnenie novej jednotky.

Táto správa sa využíva pri vytváraní novej hry, ktoré nastane po výbere ponuky „Start Match“ z menu „Multiplayer“, konkrétne v záverečnej fáze vytvárania hry. Na záver prijatia tejto správy je už všetko pripravené na započatie novej hry a tak je z tohto miesta odosielaná správa `etMatchStart`, ktorá tento stav indikuje.

3.6.2.8 Správa udalosti etObjectiveCompleted

Správa udalosti `etObjectiveCompleted` je jednoduchou správou o udalosti splnenia úlohy. Využíva sa na informovanie ostatných hráčov o splnení úlohy vybraného hráča, ktorého meno sa nachádza v atribúte správy `playername`. Klient, ktorý túto správu prijme, si zvýši vo svojej lokálnej tabuľke hráčov počítadlo splnených úloh pri vybranom hráčovi o 1.

V prípade hry pre viacerých hráčov „deathmatch“ je treba naviac zneplatniť úlohy týkajúce sa zničeného hráča vo všetkých lokálnych zoznamoch u každého hráča v hre. Správa obsahuje vo svojom atribúte `actualvalue` identifikačné číslo daného hráča a pomocou neho sú úohy v zozname identifikované a následne zneplatnené.

3.6.2.9 Správa udalosti etEnableObjective

Na obnovenie platnosti určitých úloh v zoznamoch úloh jednotlivých hráčov sa používa správa `etEnableObjective`. Pri hre viacerých hráčov „deathmatch“ sa používa táto správa na znovuobnovenie platnosti úloh týkajúcich sa hráča, ktorý si práve vytvoril nový tank. Je teda možné ďalej považovať tieto úlohy za splniteľné. Identifikácia vybraného hráča sa nachádza v atribúte správy `value`.

3.7 Zmeny v projekte TankGame

Samotný systém pre zadávanie a detekciu úloh sa nachádza v súboroch „Multiplayer.h“ a „Multiplayer.cpp“, ktoré boli pridané medzi ostatné zdrojové súbory TankGame. Systém bol implementovaný na verzii s implementovaným sieťovým kódom od Lukáša Obrdlíka.

Počas implementácie systému pre zadávanie a detekciu úloh bolo treba urobiť niekoľko zmien v stávajúcom kóde projektu TankGame. V hlavnej triede simulácie `TTankGame` bolo treba zprístupniť dátovú štruktúru `TPlayer` a atribút zoznamu hráčov `players`. Pred tieto dve položky som teda pridal kľúčové slovo `public`, ktoré týmto štruktúru a atribút sprístupní aj v metódach,

ktoré nepatria triede `TTankGame`. Ďalšou zmenou v tejto triede bolo pridanie atribútu `Match`. Tento atribút predstavuje ukazateľ na objekt triedy `TMatch`. Objekt `Match` predstavuje vlastnú hru so systémom pre zadávanie a detekciu úloh v simulácii. Detekovanie úloh vytvorenej hry sa prevádza vo funkcii `TimeTick` hlavnej triedy `TTankGame`. Do menu projektu TankGame bola pridaná položka „Multiplayer“ s ponukami „Start Match“ a „End Match“ a prislúchajúci obslužný kód pre tieto ponuky.

Do súboru „api.h“ patriacemu komunikačnému aplikačnému rozhranie sieťového kódu TankGame bola pridaná funkcia `IsServerRunning`, ktorá na základe platnosti ukazateľa `server` zisťuje, či beží v danej inštancii TankGame serverová časť.

V súbore „pdu.h“ bolo prevedených viac zmien týkajúcich sa komunikačných správ. Prvou z nich bolo pridanie nového typu `crMatch` do štruktúry `TCreateType` v správe typu `TCreatePDU`. Nadväzujúcou zmenou bolo vytvorenie nového typu správy o vytvorení objektu, ktorá sa týka vytvorenia novej hry. Nová správa je typu `TMatchCreatePDU`, odvodeného od základného typu pre správu o vytvorení objektu `TCreatePDU`.

Ďalšie zmeny sa týkajú štruktúry `TEventType` v správe typu `TEventPDU`. Bolo potrebné vytvoriť nové typy udalostných správ, preto boli do tejto štruktúry pridané hodnoty `etDeleteTank`, `etTankDeleted` a `etDeployTank` pre udalosti týkajúce sa tanku. Novými pridanými udalosťami, ktoré sa týkajú vytvorenej hry s detekciou úloh, sú udalosti `etMatchStart`, `etMatchEnd`, `etMatchDelete`, `etObjectiveCompleted` a `etEnableObjective`. Reakcie na tieto správy boli pridané do metódy `DispatchMessage` v triede `TTankGame`.

4 Záver

Hlavným cieľom bakalárskej práce bolo vytvorenie systému pre zadávanie a plnenie úloh pre projekt TankGame.

Po preštudovaní literatúry o spracovaní udalostí v aplikáciách a niekoľko článkov o vývoji sieťových počítačových hier, začal vznikať návrh systému pre zadávanie a detekovanie úloh v projekte TankGame. Nasledovalo samozrejme naštudovanie podstatných častí projektu TankGame, najmä jeho sieťovej časti. Prvá vytvorená verzia systému fungovala iba pri hre jedného hráča. Zadávanie a detekovanie úloh prebiehalo iba na jednom počítači. Bolo treba ďalej naštudovať problematiku komunikácie medzi entitami v distribuovanej simulácii TankGame. Po preniknutí do problematiky správ, ktoré si jednotlivé entity v simulácii zasielajú, vznikol návrh komunikácie pre systém detekovania úloh. Následne boli vytvorené jednotlivé správy pre entity systému. Systém následne síce fungoval pre 2 hráčov v simulácii, ale po niekoľkých testoch vyšlo najavo, že pre viacerých hráčov v simulácii funkčný už nie je. Po miernych úpravách sa však systém stal funkčným pre všetkých hráčov v distribuovanom prostredí simulácie TankGame. Na ukážku funkčnosti systému bol implementovaný mód hry „deathmatch“. Zároveň bolo treba doimplementovať niektoré metódy aplikačného rozhrania a poupraviť niektoré záležitosti týkajúce sa objektového návrhu.

Nakoniec boli prevedené testy zaťaženia systému pre zadávanie a detekciu úloh na projekt TankGame. Testy sa týkali zistenia dopadu na výkon. Neboli zistené žiadne výkonové poklesy v počte snímkov za sekundu.

Pre ďalší rozvoj systému pre zadávanie a detekciu úloh by bolo dobré doimplementovať niekoľko ďalších módov hry. Ďalej by sa dal systém využiť pre vytvorenie určitej kampane pozostávajúcej z väčšieho počtu rôznych misií. Zaujímavé by bolo spojenie tohto systému s implementovaním umelej inteligencie do projektu. Rozšírilo by to možnosti projektu o ďalšie nové možnosti.

Literatúra

[1] OBRDLÍK, L. *Síťová komunikace pro hromadnou simulaci vozidel*. Brno, 2006. Diplomová práce na Fakultě informačních technologií Vysokého učení technického v Brně. Vedoucí diplomové práce Jaroslav Kadlec.

[2] TREGLIA, D. Game Programming Gems 3. Hingham, Massachusetts: Charles River Media, 2002. ISBN 1-58450-233-9.

[3] Wikipedia:Massively multiplayer online role-playing game. [online], last modified 2007-05-09, [cit. 2007-05-10]. URL <http://en.wikipedia.org/wiki/Mmorpg>

[4] ZIMMERMAN, M., D.; ROTHSTEIN, B.; KAGANOVICH, Y. and PHAM, K. *Constructing Client-Server Multi-Player Asynchronous Networked Games Using a Single-Computer Model.*Computer Science 256-80, California Institute of Technology, Pasadena California, 1997.

Zoznam príloh

Príloha 1. CD so systémom pre zadávanie a plnenie úloh v projekte TankGame

Príloha 2. Zdrojové texty dostupné na priloženom CD v adresári `src`

Príloha 3. Programová dokumentácia dostupná na priloženom CD v adresári `docs`

Príloha 4. Návod na otestovanie demonštračného príkladu hry „deathmatch“ v súbore `manual.pdf` na CD