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

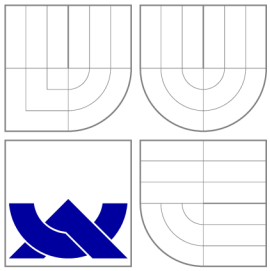
SÍŤOVÁ HRA: HON NA PONORKU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

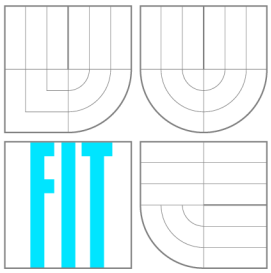
AUTOR PRÁCE
AUTHOR

DUŠAN KOVAČIČ

BRNO 2011



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

SÍŤOVÁ HRA: HON NA PONORKU

NETWORK GAME: DESTROY THE SUBMARINE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — DUŠAN KOVAČIČ
AUTHOR

VEDOUCÍ PRÁCE — Ing. IGOR SZÖKE, Ph.D.
SUPERVISOR

BRNO 2011

Abstrakt

Cílem bakalářské práce s názvem Síťová hra: Hon na ponorku bylo navrhnout a implementovat síťovou architekturu a vyřešit problémy s tím spojeny. Práce analyzuje rozdíly v použití síťových technologií a přístupu. Ve finální verzi je vytvořené dílo využívající klient-server architekturu s komplexním síťovým protokolem. Pro komunikaci mezi subjekty je ustanoveno virtuální spojení nad protokolem UDP. Minoritní část práce řeší problémy, tykající se grafické podoby aplikace a následné problémy se zobrazováním přijatých dat. Využity jsou různé frekvence zpracování a techniky, umožňující kvalitnější herní zážitek.

Abstract

The goal of this bachelor's thesis Network game: Destroy the submarine is to analyze and implement network architecture and solve all problems connected to it. In final version, the created program uses client-server network architecture. Both subjects communicate with complex network protocol. For effective data transfer, the virtual connection over UDP protocol is established. Minor part of this study solves graphical problems and problems with received data displaying. For this purpose, there are different frequencies used for data handlings and techniques that provide better game experience.

Klíčová slova

Síť, síťová hra, síťová komunikace, síťový protokol, 2D grafika, TCP, UDP, klient, server, predikce vstupu.

Keywords

Network, network game, network communication, network protocol, 2D graphics, TCP, UDP, client, server, input prediction.

Citace

Dušan Kovačič: Síťová hra: Hon na ponorku, bakalářská práce, Brno, FIT VUT v Brně, 2011

Síťová hra: Hon na ponorku

Prohlášení

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

.....

Dušan Kovačič

25. července 2011

Poděkování

Chcel by som poďakovať vedúcemu práce Ing. Igorovi Szőkemu Ph. D., za vedenie projektu a poskytnutú spätnú väzbu. Ďalej by som chcel poďakovať spoločnosti Safetica Techologies s.r.o., v ktorej som posledný rok pracoval ako programátor, čo mi rapídne zvýšilo schopnosť navrhovať a a programovať projekty. Ďakujem všetkým testerom, ktorý sa podieľali na testovaní, ako aj diskutujúcim na fóre o vývoji hier s názvom *www.gamedev.net* [1]. Aj ich príspevky boli inšpiráciou k mojej ďalšej činnosti.

Obsah

Kapitola 1

Úvod

Vo všeobecnosti je tvorba hier v súčasnej dobe vedou a často zamestnáva obrovské tímy vývojárov s rozpočtami presahujúcimi desiatky miliónov dolárov. Ale nebolo tomu vždy tak. História tvorby počítačových hier siaha do 40.tych rokov 20. storočia, kedy vznikali prvé pokusy o vytvorenie akejsi zábavy, na dovtedy len pracovnom nástroji - počítači. Na druhej strane išlo však iba o pokusy vytvoriť niečo jednoduché, zabávajúce maximálne jedného aktívneho hráča. V týchto dobách bola celá počítačová technológia ešte v plienkach a vytvorené výsledky boli viac dielom jednotlivca ako väčších tímov.

Zlom v tvorbe sieťových hier prišiel s rozvojom internetu. Hlavným limitujúcim faktorom však bola veľká odozva sieťového pripojenia, ktorá neumožňovala priebeh a simuláciu prostredia v reálnom čase. Všetko však zlepšil nástup rýchlejších spojení v neskorších 90. tych rokoch. Výsledné hodnoty odozvy sa zrazu čoraz viac blížili k 50 milisekundám nových káblových pripojení. Oproti dovtedy majoritnou skupinou ľudí využívajúcich modemov to bol obrovský skok, ktorý zrazu otvoril nové, neprebádané možnosti - tvorbu sieťových hier.

Prvou sieťovou hrou, pracujúcou v reálnom čase, bol dnes už legendárne známy Quake. Hlavne zásluhou Johna Carmacka, vedúceho tímu vývojárov spoločnosti ID Software, ktorý sa stal postavou nasledujúcich desiatich rokov herného priemyslu, je dnes tvorba počítačových hier na tak vysokej úrovni, keď požiadavky softvéru neprestajne prevyšujú možnosti hardvéru. Bola to práve táto hra, ktorá priniesla revolúciu v tvorbe sieťových protokolov a klient-server komunikácie. Zároveň sa však dokázala vyrovnať s existujúcimi odozvami a poskytla skvelý herný zážitok.

1.1 Sieťová hra pre viac hráčov

Sieťové hra pre viac hráčov inak nazývaná aj multiplayer hra je taká videohra, v ktorej môže naraz v jednom hernom prostredí súťažiť viac hráčov v rovnakom čase. Na rozdiel od hier pre jedného hráča (singleplayer), kde človek vyzýva naprogramovaných, AI[1] kontrolovaných oponentov, ktorý často postrádajú štipku umu a cieľavedomého ľudského myslenia, sa hráč stretáva so seberovným protihráčom a hra tak naberá úplne novú dimenziu.

Multiplayer hry umožňujú hráčom využitie vzájomnej spolupráce a interakcie napríklad vo forme partnerstva v jednom hernom tíme, súťaže alebo iného vzájomného vyzývania. Poskytujú určitú formu spoločenskej komunikácie, ktorá takmer vždy v hrách orientovaných pre jedného hráča chýba. V rôznych druhoch typov multiplayer počítačových hier, môžu hráči vzájomne súťažiť, spolupracovať s ľudským partnerom v záujme dosiahnuť spoločný

[1] umelá inteligencia

cieľ, dohliadať na aktivitu iných hráčov alebo sa zapojiť do hier, ktoré obsahujú všetky možné vyššie uvedené kombinácie.

Pre lepšie pochopenie podstaty hier pre viac hráčov je nutné porozumieť rozličným herným typom, ako napríklad deathmatch[2], team deathmatch[3], MMORPG[4] asociované s rozličnými formami PvP[5] a týmových PvE[6] bojov, CTF[7], domination[8], co-op[9] a rôznych iných typoch založených na splnení určitých misií, často pozostávajúcich z dobytia/obránenia častí základní a podobne. Multiplayer hry typicky vyžadujú od hráčov, aby využívali zdroje jedného systému (viacerí hráči hrajú na jednom počítači) alebo sieťové technológie, ktoré umožňujú komunikáciu naprieč sieťou a internetom.

1.2 Ciele projektu

Cieľom tohto projektu je vytvoriť jednoduchú sieťovú hru s herným módom typu deathmatch, dvojrozmernou grafikou a prepracovanou komunikáciou, poskytujúcu zábavu viac hráčom, nie však rozsahom komerčného projektu, čo by nebolo z časových dôvodov možné. Jeho snahou je príkladne demonštrovať kvalitný sieťový protokol a riešiť základné problémy súvisiace s problematikou.

Základným kameňom pre sieťovú hru je teda sieťový protokol, ktorý je využívaný logikou aplikácie pre výmenu dát medzi klientom a serverom. Efektívne implementuje prenos herných správ na základe stanovených požiadaviek. Prenášané dáta sa postupne spracujú na programové štruktúry a následne využívajú pri výpočte a riadení herných mechanizmov. Herný systém je rozdelený na dva samostatné podprogramy využívajúc klient-server architektúru. Klient zabezpečuje spracovanie užívateľovho vstupu, zobrazenie a prezentáciu grafiky, čiastočné výpočty polohy hráčom kontrolovaných objektov. Získané dáta analyzuje, spracuje a posiela na server.

Server je druhým samostatným podprogramom výsledného systému, ktorý slúži predovšetkým ako centrálny autoritatívny bod, ktorý poskytuje konektivitu a synchronizáciu klientov. Jeho úlohovou je najmä spracovávať a obsluhovať prichádzajúce dáta. Vypočítavať kolízie objektov a upravovať súradnice vznikajúce pri chybných výpočtoch na strane klienta je taktiež výhradne doménou servera.

Cieľom práce je dosiahnuť správne fungovanie sieťovej hry pre viac hráčov s názvom: Hon na ponorku, objasniť štruktúry programu, odpovedať na základné otázky a vysvetliť problémy súvisiace s danou problematikou. Základným a najdôležitejším cieľom je poskytnúť čitateľom jasný a názorne vyriešený problém sieťovej komunikácie, synchronizácie servera a klienta. Vysvetliť princíp fungovania celého riešenia, popísať sieťový protokol, jeho výhody, nevýhody, možné zlepšenia a následne ujasniť celkový princíp funkčnosti.

[2] herný mód, všetci hráči bojujú proti sebe
[3] herný mód, hráči tvoria tímy, ktoré bojujú proti sebe
[4] typ hry, v ktorej je naraz v jednom hernom svet aktívnych tisíce hráčov
[5] skratka player versus player, hráči bojujú proti sebe
[6] skratka player versus enviroment, hráči bojujú spolu proti počítačom kontrolovaným oponentom
[7] herný mód, v ktorom hráč stráži vlajku svojho tímu a snaží sa získať vlajku oponentov
[8] herný mód, úlohou hráča je spolu s tímom kontrolovať čo najväčšie možné územie na mape
[9] herný mód, v ktorom hráč spolupracuje s ľudským partnerom a súťaží proti počítaču

Pri tvorbe masívnejších projektov je nutné dodržať základnú formu tvorenia programu. Návrh riešenia bol rozdelený na tri časti a to:

- návrh sieťového protokolu,
- návrh štruktúry klienta,
- návrhu štruktúry servera.

Nasledovala implementácia, pozostávajúca z:

- voľby vývojového prostredia,
- implementácie sieťového protokolu,
- implementácia klienta,
- implementácia servera.

Finálny výsledok bol podrobený testovaniu a ladeniu na rôznych systémoch.

1.3 Prehľad práce

Kapitola 2 informuje o základných možnostiach hry, popisuje ciele hry, informuje užívateľa o nutných systémových úpravách pre správne fungovanie, objasňuje základné ovládanie a zobrazuje konečný počet stavov hráča, v ktorých sa môže počas hrania hry vyskytnúť. Nakoniec popisuje grafické užívateľské rozhranie, systémové nároky.

V kapitole 3 je obsiahnutý a podrobne vysvetlený výsledný návrh sieťového protokolu, klienta a servera. Diskutuje výhody a nevýhody jednotlivých riešení a zdôvodňuje konkrétny výber použitých knižníc a techník.

Kapitola 4 sa zaoberá implementačnými problémami a ich riešeniami. Vysvetľuje voľbu programovacieho jazyka, využité vývojové prostredie a dôvod jeho použitia. Následne opisuje implementáciu sieťového protokolu, vysvetľuje použitie konkrétnych programových štruktúr a graficky prezentuje. Ďalej sa zaoberá implementáciou klienta a servera a rieši s tým spojené problémy.

Kapitola 5 informuje o ladení a testovaní výsledného projektu a jeho konkrétnych častí. Graficky zobrazuje využitie sieťových a výpočtových zdrojov.

Kapitola 6 zhodnocuje celý projekt, jeho úspešnosť a načrtá možný budúci vývoj. Táto práca nadväzuje na semestrálny projekt vypracovaný v 5. semestri štúdia, z ktorého bola prevzatá a upravená časť návrhu a implementácie.

Kapitola 2

Definícia hry

Sieťová hra Hon na ponorku je klasickou jednoduchou arkádovou hrou, ktorá využíva dvojrozmernú grafiku. Hru môžu naraz hrať dvaja až štyria hráči. Pre jej spustenie je potrebná lokálna sieť. Úlohou hráča je eliminovať protivníkov použitím zbraní. Hráč sa pohybuje po mape prostredníctvom klávesnice, pričom sa vyhýba nepriateľským strelám a snaží sa zneškodniť oponenta. Pri spustení sa hráčovi zobrazí základné jednoduché grafické menu, ktoré ho informuje o nutnosti zadať IP adresu herného servera. Následne je hráč pripojený do hry, vygeneruje sa mu náhodná pozícia spawnu[1] a môže začať hrať. Stlačením zvolených kláves udáva smer a rýchlosť pohybu po mape, a strieľa navádzané rakety, ktorými sa snaží zneškodniť ponorky oponentov. V prípade úspešného zásahu je hráč ocenený. V prípade, kedy hráča zasiahne nepriateľské torpédo, čaká stanovený čas a následne je mu vygenerovaná nová východzia pozícia.

2.1 Cieľ hry

Základným cieľom hry je eliminovať ostatných hráčov použitím navádzaných torpéd. Pri zásahu oponenta je hráčovi pripísaný bod. Úlohou hráča je pohybovať sa tak, aby sa vyhol nepriateľským ponorkám a torpédam a snažiť sa pomocou svojich torpéd zasiahnuť a eliminovať ostatných. Vyhráva hráč, ktorý dosiahne najvyššie bodové ohodnotenie v stanovenom hracom čase. Hráč taktiež nesmie naraziť do prekážky, ani do oponenta.

2.2 Použitie hry

Herný systém je rozdelený na dve samostatné časti: prvou je klient s názvom súboru SMClient.exe, ktorý zabezpečuje interakciu s hráčom, prezentáciu grafického výstupu a príjem príkazov. Druhou časťou je server s názvom spustiteľného súboru SMServer.exe, zabezpečujúci spracovanie dát a synchronizáciu.

Pre spustenie hry a jej správnu funkčnosť je potrebné dodržať nasledovný postup. Ako prvé je nutné správne nakonfiguovať firewall. Pre serverovú komunikáciu je potrebné otvoriť port TCP s číslom 27015, čo je základný port pre prijímanie správ, a rozsah portov 27100 až 27108 UDP pre komunikáciu herných mechanizmov a úspešné prenášanie dát. Pre umožnenie komunikácie klienta je nutné odblokovať porty UDP v rozsahu 20000 až 20100. V systémoch Windows XP a novších je možné túto zmenu uskutočniť veľmi jednoducho a

[1] objavenie hráča na novej pozícii v prípade začiatku hry alebo zničenia

to povolením výnimky v nastaveniach firewallu pre program SMClient.exe, prípadne SMServer.exe. Ďalšou požiadavkou klienta je z dôvodu grafického výstupu, potrebná inštalácia DirectX 9 runtime prostredia.

V prípade úspešného prevedenia systémových zmien stačí spustiť program SMServer.exe na počítači s úlohou servera a následne program SMClient.exe. Po uskutočnení týchto krokov je hra schopná prevádzky.

Po spustení klienta sa zobrazí grafické menu, popísané na obrázku číslo 2. Do zobrazeného editboxu je nutné vložiť IP adresu počítača, na ktorom beží program server. Následne stačí stlačiť klávesu enter alebo tlačidlo „connect“a hra sa môže začať.

2.3 Základné ovládanie

Tabulka 2.1: Základné ovládanie

Akcia	Kláves
Zobrazenie skóre	Tab
Streľba	Medzera
Pohyb	**Kláves**
Dopredu	Šípka dopredu
Dozadu	Šípka dozadu
Doľava	Šípka doľava
Doprava	Šípka doprava

2.4 Popis hry

V nasledujúcich odstavcoch je popísaná základná funkčnosť hry, jej princípy a charakteristiky.

2.4.1 Fyzika

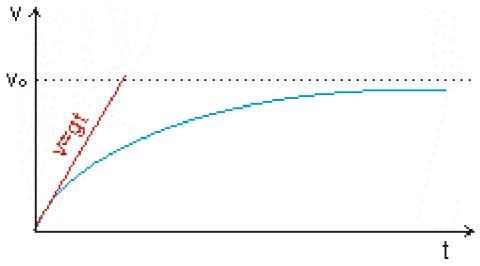
Pre pohyb telies vo virtuálnom prostredí hry, je vhodné aby objekty aspoň voľne aproximovali správanie v reálnych situáciách. Výsledný efekt dobre naprogramovanej fyziky skvalitňuje herný zážitok a dodáva hre ten správny efekt pri ktorom hráč iba neveriacky krúti hlavou, kam sa až úroveň súčasných hier dostala. V tomto projekte bude využitá základná fyzikálna aproximácia pohybu telies v kvapaline.

Pohyb ponorky

Pohyb ponorky v hernom prostredí sa zakladá na princípe pohybu objektov v kvapaline, kde je do úvahy braný odpor prostredia proti smeru pohybu. Gravitačná a vztlaková sila sú si rovné a teda nemajú vplyv na výsledný pohyb. Pri aproximovaní je použitý základný Stokesov vzorec:

$$F_0 = -kv$$

Tento fyzikálny vzorec vyjadruje silu odporu prostredia voči telesu v aktuálnej rýchlosti v. Konštanta k je súčinom vlastností kvapaliny a vlastností telesa. Do úvahy nie sú brané prípadné víry, ktoré by mohli vzniknúť za telesom. Výslednú funkciu pohybu tak znázorňuje obrázok č. 2.1. Princíp pohybu bol prebratý a analyzovaný z *Pohyb telesa vo viskóznej tekutine* [6].



Obrázek 2.1: Zrýchlenie ponorky. Na osi v je označená rýchlosť, na osi t čas. Modrá krivka znázorňuje nárast rýchlosti v kvapaline za čas, červená úsečka slúži pre porovnanie a spolu so vzorcom označuje nárast rýchlosti v čase pri voľnom páde. V_0 informuje o maximálnej rýchlosti, ktorú môže teleso dosiahnuť. Je zrejmé, že odvodená závislosť rýchlosti od času v kvapaline má v porovnaní so závislosťou pre rýchlosť pri voľnom páde, keď na teleso pôsobí len tiažová sila, zložitejší priebeh. Obrázok bol prebratý z [6].

Ako je možné pozorovať, zrýchlenie telesa je možné vyjadriť exponenciálne, až kým nedosiahne meznú rýchlosť, ktorá má limitu vo V_0 . Následne sa zrýchľovanie zastaví a pohyb telesa možno charakterizovať ako rovnomerný priamočiary s konštantným odporom voči smeru pohybu. Implementácia tohto princípu v hre je aproximačná a môže sa v určitých prípadoch výrazne odlišovať od reality.

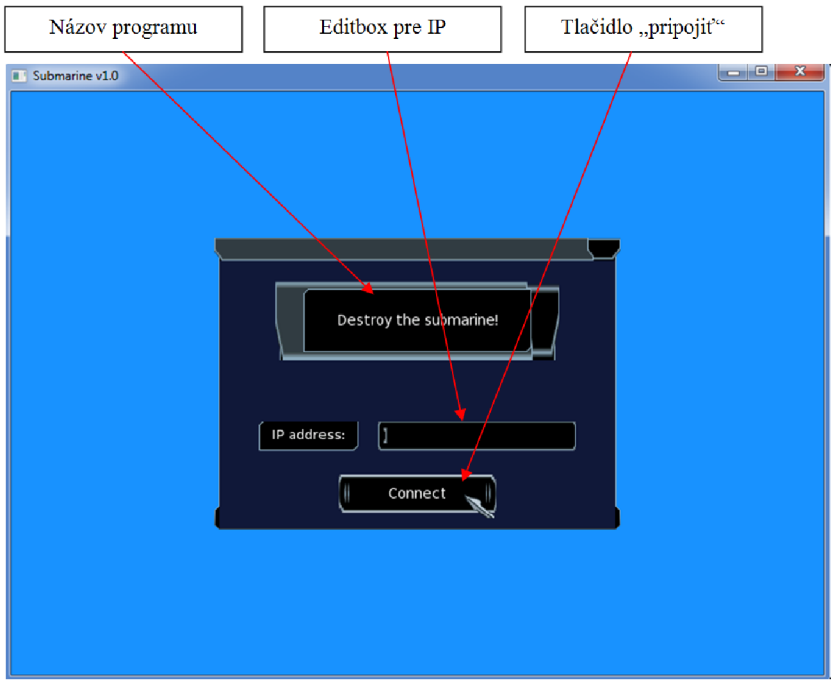
Navádzanie torpéd

V dnešnom svete armádnych technológií sú všetky torpéda navádzané a keďže hra Hon na ponorku sa snaží vychádzať z reality, nie je možné túto skutočnosť zanedbať. Každá ponorka má informácie o okolitých objektoch a teda môže torpéda automaticky smerovať na najbližší cieľ. V prípade vystrelenia určitým smerom torpédo analyzuje objekty na mape a zistí vzdialenosť k najbližšiemu cieľu. Následne upraví svoj smer. Každé torpédo je však limitované objemom pohybu, ktorý môže vykonať pri jednom výpočte smerovania, takže nie je možné, aby malo účinnosť zásahu 100% a hra by bola nehrateľná. Torpédo je navádzané a môže sa trafiť, ale aj nemusí, v prípade kedy hráč vie, ako sa mu vyhnúť. Tento mechanizmus dodáva hre celkom dobrú zábavnosť.

2.4.2 Popis grafického užívateľského rozhrania

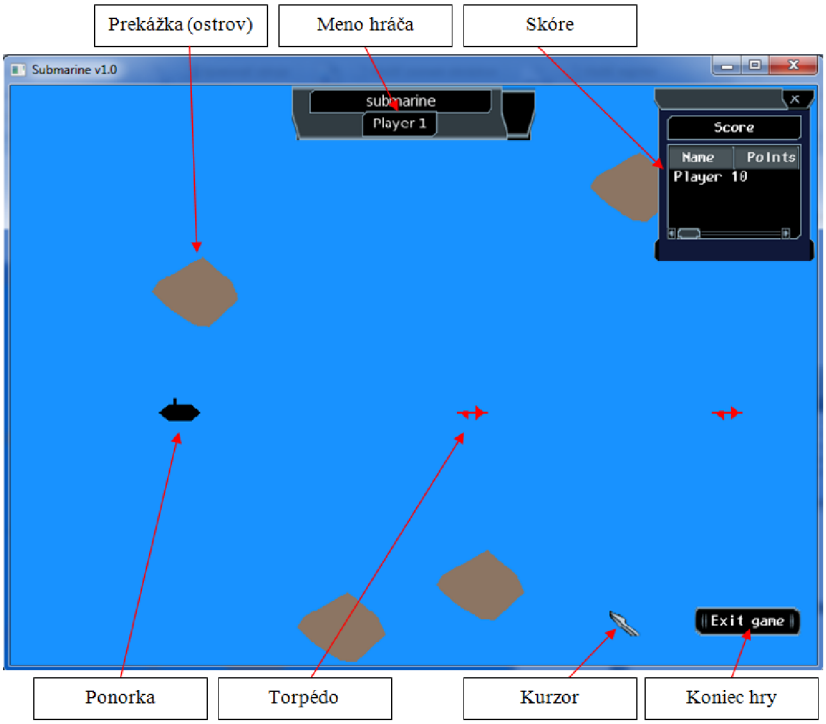
V sieťovej hre Hon na ponorku je možné stretnúť sa s dvoma typmi grafického užívateľského rozhrania. Po počiatočnom spustení je zobrazené menu, ktoré je možné vidieť na obrázku č. 2.2 a následne v samotnej hre je druhý typ, popisujúci aktuálnu situáciu na bojisku, ohodnotenie jednotlivých hráčov, popísaný na obrázku č. 2.3.

Menu



Obrázek 2.2: Menu. Obrázok znázorňuje úvodné menu, ktoré sa zobrazí pri štarte klienta.

Herné GUI



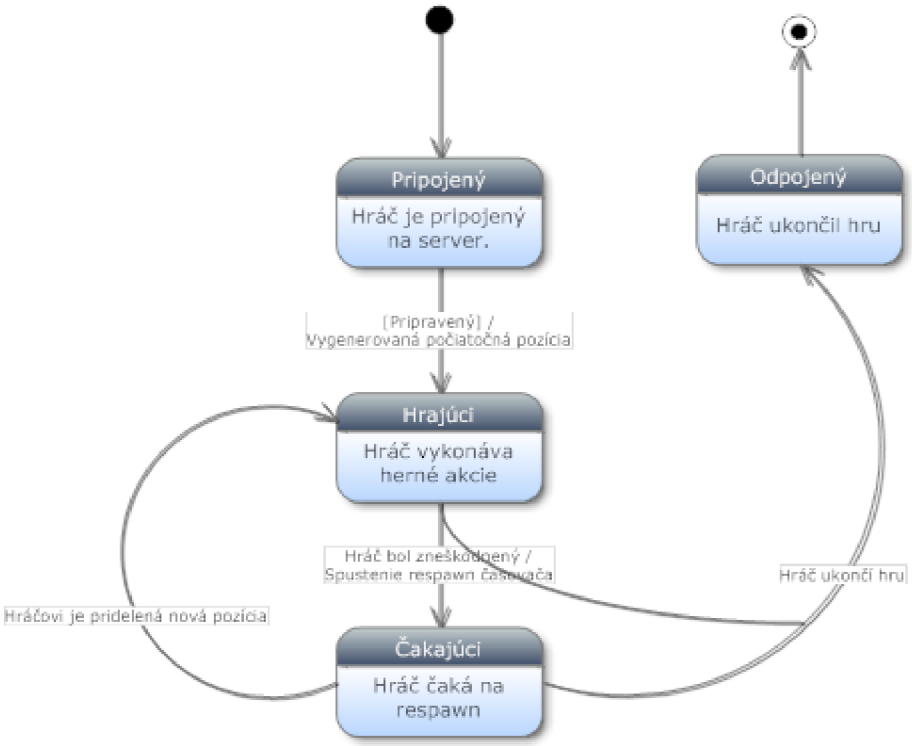
Obrázek 2.3: Herné grafické užívateľské rozhranie. Na obrázku je znázornené GUI, ktoré sa zobrazí pri spustení hry.

2.5 Nároky na software a výpočtové zdroje

Program sieťová hra Hon na ponorku je nenáročný na výkon a pre svoj beh mu dokonale postačuje hardwarová konfigurácia dnes každého kancelárskeho počítača na ktorom je možné spustiť aspoň Windows XP. Nevyužíva žiadne časovo náročné výpočty, takže nezaťažuje operačný systém a nevyužíva procesor viac ako piatimi percentami výkonu. Taktiež je nutné, aby počítač obsahoval sieťovú kartu a bol pripojený k lokálnej sieti.

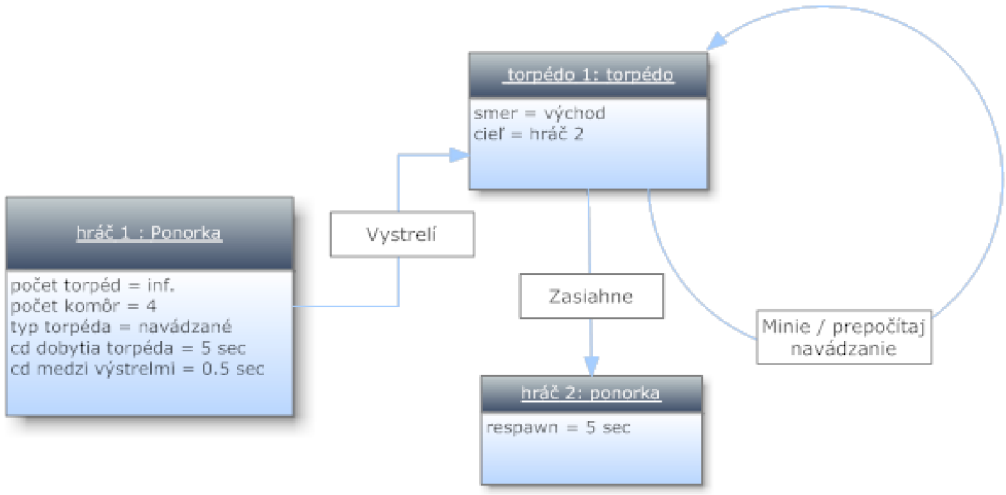
V prípade softvérových požiadaviek sa jedná o nutnosť vlastnenia operačného systému platformy Windows vo verzii XP alebo novšej, na ktorej je nutné mať nainštalované DirectX 9 runtime prostredie. Kompatibilita so staršími systémami nie je možná, keďže spomínaná grafická knižnica nepodporuje staršie systémy.

2.6 Konečný automat stavov hráča - stavový diagram



Obrázek 2.4: Stavový diagram hráča. Obrázok popisuje herné stavy, v ktorých sa hráč môže počas hrania vyskytnúť.

2.7 Popis jednotlivých prvkov hry a ich správania



Obrázek 2.5: Správanie objektov v hre. Obrázok predstavuje formálny popis hry, informuje o časovaniach jednotlivých herných mechanizmov. Skratka cd označuje cooldown[3].

[3]čas, ktorý musí uplynúť, aby udalosť mohla znovu nastať

Kapitola 3

Návrh

V počiatočnom štádiu tvorby programu, ako je sieťová hra, je nutné ujasniť si požadovaný cieľ a získať celkovú víziu budúceho výsledného programu. Tento virtuálny obraz výsledku je jednoducho dosiahnuteľný preštudovaním existujúcich projektov rôznych komerčných firiem, zaoberajúcich sa činnosťou vývoja hier. Ukážkovým príkladom je spoločnosť Valve Corporation, dlhé roky vyvíjajúca špičkové komerčné, celosvetovo uznávané projekty ako Half-life, Counter-strike, a pod. Na svojej vývojárskej stránke, podporujúcej tvorbu nových hier pomocou vytvoreného Source engine API[1], dôkladne dokumentuje a vysvetľuje princíp fungovania ich konkrétneho riešenia. Dlhoročná prax a trh overili, že na túto firmu je naozaj možné sa spoľahnúť. Základné princípy fungovania, riešenia a skúsenosti z tvorby ich enginu sú aplikovateľné aj v tomto projekte a boli využité pre implementáciu v hre.

3.1 Výber sieťového modelu

Pri tvorbe distribuovaných systémov, ku ktorým môžeme s určitosťou zaradiť aj sieťové hry, je nutné špecifikovať úlohu každej časti a spôsob ich vzájomnej komunikácie. V praxi používané a principiálne ľahko pochopiteľné sú dva sieťové modely. Peer-to-peer inak označovaný aj ako P2P a model Klient-server.

V Peer-to-Peer modeli majú všetky prvky systému rovnakú funkčnosť, neobsahuje klientov, ani servery, iba rovnocenné sieťové uzly, ktoré plnia voči iným uzlom v sieti úlohu servera aj klienta súčasne. Hlavne touto vlastnosťou sa P2P sieťový model výrazne odlišuje od modelu klient-server, kde výmena dát zvyčajne prebieha cez centrálny server. Pri komunikácii typu Peer-to-Peer si každý peer lokálne uchováva informácie o ostatných uzloch, vyžaduje však väčšie množstvo prenesných dát ako je tomu v prípade klient-server. Má však výhodu, že centrálny bod - server nie je potrebný. Rozdiely, výhody a nevýhody použitia sú podrobnejšie popísané v knihe *Multiplayer game programming* [3].

Po analyzovaní vlastností oboch možností som dospel k záveru, že vhodným riešením bude využiť model klient-server. Jeho dve časti možno charakterizovať nasledovne.

Základné vlastnosti klienta sú:

- iniciuje požiadavky
- čaká a prijíma odpovede
- zvyčajne sa pripája na malé množstvo serverov v jeden čas

[1] skratka pre application programming interface

- komunikuje s užívateľmi väčšinou prostredníctvom grafického užívateľského rozhrania

Charakteristiky servera:

- pasívny
- čaká na požiadavky od klienta
- prijíma požiadavky, spracúva ich a následne odpovedá
- zvyčajne akceptuje pripojenia od väčšieho množstva klientov
- typicky sa priamo neovplyvňuje s koncovými užívateľmi

V projekte sieťovej hry Hon na ponorku časť klient zabezpečuje ustanovenie spojenia, interakciu s užívateľom a základné výpočty. Úlohou servera je predovšetkým analyzovať vzniknuté situácie, vyvodzovať následky, a následne informovať klientov o aktuálnom stave.

3.2 Návrh sieťového protokolu

Pre správne, efektívne a na sieť nenáročné fungovanie sieťovej aplikácie je nutné vypracovať dôkladný sieťový protokol, minimalizujúci prenášané dáta a maximalizujúci rýchlosť. Keďže sieťový protokol je súčasťou herného enginu, ktorý je cieľom vytvoriť v tejto práci, nie je vhodné použiť žiadne z existujúcich riešení. Bolo by možné postaviť celú hru na už vytvorených riešeniach, ktoré boli vyvinuté komerčnými firmami, ako napríklad Source engine pre Half-Life 2, alebo Quake 3 engine. To však nie je cieľom, lebo v prípade použitia jedného zo spomenutých enginov by bolo pužívané iba poskytnuté API a celá realizácia ostane skrytá. Ostáva teda len jedna možnosť, a to navrhnúť a vytvoriť vlastný komunikačný mechanizmus inšpirovaný spomenutými komerčnými projektmi.

Pre správnu predstavu o budúcom protokole bolo potrebné preštudovať viaceré odborné články, návody a dokumentácie zaoberajúce sa touto problematikou. Konkrétne bol inšpiratívny najmä článok od autorov hry Age of Empires s názvom *1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond* [4], vývojárska stránka spoločnosti Valve poskytujúca dokumentáciu k ich hernému enginu *developer.valvesoftware.com* [2] a mnoho útržkovitých článkov ohľadne Quake 3 sieťového protokolu. Po dôkladnom zhodnotení nadobudnutých informácii je dilema jasná, a to aký vhodný protokol na transportnej vrstve OSI modelu z balíka internetových protokolov použiť. Možnosti sú dve, a to protokol TCP alebo UDP.

3.2.1 Protokol TCP

Protokol TCP patrí do základného balíka internetových protokolov na transportnej vrstve OSI modelu. Hlavnou charakteristikou je orientácia na spojenie. V súčasnosti je zdokumentovaný v IETF RFC 793 [12]. Funkčnosť TCP pozostáva v zabezpečení strednej vrstvy medzi IP protokolom na sieťovej vrstve a aplikáciami. Základnými vlastnosťami sú:

- založené na spojení
- garantuje spoľahlivý prenos dát s packetmi v správnom poradí
- automaticky rozdeľuje dáta do packetov

- kontrola toku
- jednoduché použitie

TCP sa typicky používa napríklad pri aplikačných protokoloch http, smtp, ssh.

3.2.2 Protokol UDP

User Datagram Protocol (UDP) je tzv. nespoľahlivý protokol z balíka internetových protokolov. UDP protokol prenáša datagramy medzi počítačmi v sieti, ale na rozdiel od TCP nezaručuje, že prenášaný paket sa nestratí, že sa nezmení poradie paketov, ani že sa niektorý paket nedoručí viackrát. Vďaka tomu je UDP pre ľahké a časovo citlivé účely rýchlejší a efektívnejší. Jeho bezstavová povaha je tiež užitočná pre servery, ktoré odpovedajú na malé požiadavky mnohých klientov.

UDP sa používa napríklad pre DNS, streamované médiá, prenos hlasu alebo videa (VoIP) a online hry. Protokol je zdokumentovaný v IETF RFC 768 [11].

Základné charakteristiky UDP:

- žiadny koncept spojenia
- žiadna záruka spoľahlivosti alebo poradia packetov, môžu byť doručené mimo poradia, prípadne duplikáty alebo nemusia byť doručené vôbec
- potrebné manuálne rozdelenie dát na packety
- nutná detekcia straty packetov a následné riešenie

3.2.3 Výber protokolu

Výber protokolu sa zdá na prvý pohľad jasný. TCP zabezpečuje všetko to, čo aplikácia potrebuje a jeho použitie je triviálne. Naproti tomu UDP neposkytuje v podstate žiadne služby a je nutné všetko implementovať samostatne. Zo spomínaných argumentov by sa zdalo, že voľba je samozrejmá, a to protokol TCP. Vzniknutá úvaha však nie je správna.

Použitie TCP je síce možné, ale určite nie je optimálne. Z princípu fungovania protokolu TCP je zrejmé, že nie všetky mechanizmy, ktoré sú využívané pre kontrolu toku a spoľahlivý prenos dát sú vhodné pre typ aplikácie sieťová hra. Konkrétne je problematický najmä algoritmus sliding window vo variante GO-BACK-N. Existujú síce aj komerčné projekty, využívajúce hlavne TCP komunikáciu, ale to iba v prípade, ak sa jedná o hru, v ktorej odozva nie je limitujúci faktor. Sú to predovšetkým ťahové hry, poprípade stratégie.

V sieťových hrách pre viac hráčov, vyžadujúcich simulácie v reálnom čase, by nemal byť použitý protokol TCP pre kontrolu herného mechanizmu. Pre ujasnenie dôvodu, prečo sú dané vlastnosti nevhodné, je nutné názorne zjednodušiť fungovanie celého algoritmu spoľahlivého doručovania správ TCP na triviálnu úroveň.

Problém s využívaním protokolu TCP súvisí s princípom jeho fungovania. V hrách, na rozdiel od webových prehliadačov, emailov alebo väčšiny ostatných aplikácii, máme požiadavku na doručenie packetov v reálnom čase. Pre mnohé časti hry, ako napríklad užívateľský vstup alebo pre aktuálnu pozíciu postáv vôbec nezáleží, čo sa stalo o sekundu späť. Jediné, čo je podstatné, sú aktuálne platné dáta.

Pre lepšie pochopenie zvážme veľmi jednoduchý príklad sieťovej hry typu „strieľačka“. V princípe naozaj nikdy nie je dobré, aby to fungovalo nasledujúcim spôsobom: Každý jeden krát, čo klient zašle akciu na server (napríklad: stlačenie klávesy, pohyb myši, alebo iného

ovládača) a každý jeden obraz aktuálneho sveta, čo server vypočíta z daného vstupu pre každého hráča, je zostavený packet a poslaný sieťou. Ako náhle je stratený, všetko sa na malú chvíľku zastaví a čaká sa znova na odoslanie daného packetu. Na strane klienta objekty prestanú prijímať aktuálne informácie, a tak že sa zdá, akoby hra stála. Na strane servera sa prestane spracúvať vstup od užívateľov, hráči sa nemôžu hýbať, strieľať alebo vykonávať akékoľvek iné akcie. Keď stratený packet konečne dorazí, server dostane informáciu, ktorá už vôbec nemusí byť aktuálna, ktorá nás už teda ani nezaujíma, plus packety, ktoré stáli v rade na odoslanie, sú poslané v jednom frame, tak že dáta sú zrazu nakopené na seba. Vyššie uvedené úvahy boli prebraté z *Glenn Fiedler's Game Development Articles and Tutorials* [5] a dôkladne analyzované.

Z vyššie uvedeného vyplýva, akú cenu je potrebné zaplatiť za to, aby vytvárané spojenia zaručovali úplný a zoradený prenos packetov.

V prípade sieťovej hry nie je prioritou spoľahlivý prenos, zabezpečujúci poradie packetov. Prvoradou požiadavkou je dostať dáta na server najrýchlejšie, ako je to možné, bez potreby čakať na znovu odoslanie stratených dát. To je hlavný dôvod, prečo je dobré v majoritnej väčšine prípadov využiť protokol UDP. Informácie o funkčnosti oboch protokolov boli prevzaté z *TCP/IP Tutorial and Technical Overview* [10].

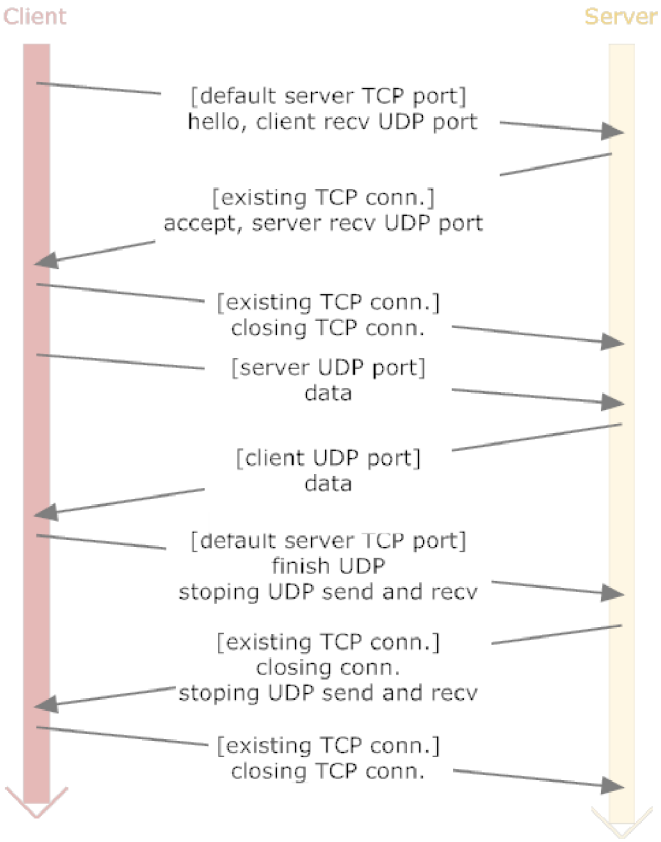
Protokol TCP je v herných enginoch využívaný hlavne pri správach, ktoré nie sú priamo súčasťou herného mechanizmu a je nutná absolútna garancia ich správneho doručenia. Chceme predovšetkým využiť výhody TCP, hlavne spoľahlivý prenos dát. Správy o pripojenie sa na server, vyžiadanie portov pre komunikáciu a podobné riešenia technického charakteru je vhodné implementovať nad spomínaným protokolom. Pri aplikácii týchto zistení vo vzťahu k vývoju programu je základom sieťovej komunikácie klienta použitý protokol TCP a následne, pre odosielanie správ zohľadňujúcich stav hry, protokol UDP, hlavne pre pohyb a rôzne iné činnosti, pre ktoré je potrebné preniesť veľký objem správ s malou veľkosťou. Výsledný pomer TCP vs. UDP je zreteľne v prospech UDP, keďže výhradne pomocou neho prebieha všetká dátová komunikácia, čo sa týka prenosu herných správ. Výhody protokolu UDP sú v tomto prípade zrejmé a v konečnom dôsledku poskytuje podstatne rýchlejšiu odozvu servera ako protokol TCP.

3.2.4 Výsledný návrh komunikácie

Na obrázku číslo 3.1 je znázornený výsledný protokol. Ako je možné pozorovať, prvý iniciuje spojenie klientov a zasiela požiadavku o vytvorenie nového herného spojenia s „hello“packetom cez protokol TCP, ktorá obsahuje číslo UDP portu, na ktorom bude klient prijímať UDP správy. Následne server potvrdí cez vytvorené TCP spojenie danú udalosť packetom „accept“, ktorý obsahuje číslo protokolu, na ktorom bude server prijímať dátové správy iba od tohto konkrétneho klienta. Tým pádom sú správy od každého klienta identifikovateľné pomocou čísla portu. Následne klient TCP spojenie ukončí a začne sa komunikovať výhradne pomocou UDP. V prípade, kedy má klient potrebu sa odpojiť, opätovne otvorí TCP spojenie a zašle packet „finish UDP“, po ktorom sú obe „virtuálne“spojenia ukončené.

3.2.5 Binárny vs. textový prenos dát

Posledným problémom, o ktorom je nutné diskutovať pri návrhu protokolu je forma prenosu dát. Na výber sa črtajú dve riešenia. Jednou z možností je posielať dáta v textovej forme, kedy je protokol potom označovaný ako „plain text protocol“. Výhody textového protokolu spočívajú najmä v ľahkej pochopiteľnosti a intuitívnosti, pretože všetky dáta sú prenášané v ľudsky čitateľnej podobe. Typickým príkladom, s rozšírením širokopásmového internetu



Obrázek 3.1: Komunikačný protokol. Obrázok znázorňuje ustanovenie spojenia a prenos dát medzi klientom a serverom.

v dnešnej dobe často používaným, je prenos XML. Toto riešenie má však nevýhodu veľkého objemu dát v správach, náročného spracovania a tým pádom vyššieho zaťaženia siete ako aj výpočtových zdrojov. Je využívané predovšetkým v protokoloch, ktoré sú súčasťou otvorených systémov a je veľký predpoklad, že v budúcnosti budú implementované v rôznych typoch klientov.

Na druhej strane existuje možnosť posielať dáta v binárnej forme, čo je podstatne rýchlejšie na spracovanie aj prenos. Hlavnou nevýhodou je značná zložitosť. Každý jeden bit v správe má totiž svoj špecifický význam a jeho zmena podstatne ovplyvňuje činnosť príjemcu. Binárny typ protokolov je využívaný predovšetkým v uzavretých systémoch náročných na rýchlosť, čo je presne prípad projektu sieťovej hry. Z týchto dôvdov bude výsledný protokol binárny.

3.3 Návrh servera

Server je časťou klient-server modelu, ktorá má na starosti hlavne sieťovú komunikáciu, synchronizáciu a redukciu lagu, vznikajúceho prenosom dát v rámci siete. Využíva rôzne techniky, aby konečný výsledok vynikal hrateľnosťou. Dobre naprogramovaný server je tou najdôležitejšou častou sieťovej hry. Môže byť súčasťou konečného programu, poskytnutého bežnému užívateľovi, využiteľný v prípade LAN hry alebo je samostatným programom, pomocou ktorého je možné vytvoriť takzvaný dedikovaný server, využívaný najme pri in-

ternetových hrách.

Základnou požiadavkou na server je schopnosť obsluhovať dostatočný počet pripojených klientov a zároveň prijímať nové prichádzajúce spojenia.

Ako bolo spomenuté, sever môže mať dve formy z hľadiska celistvosti herného systému. Prvou možnosťou je, že je súčasťou jedného programu spolu s klientom. Tento variant je vhodný pri krátkodobých hrách na lokálnej sieti, kde samotný užívateľ počítača, na ktorom je server spustený, sa aktívne zapája do hry. Výhodou z toho prameniacou je jednoduchšia spustiteľnosť hry. Na druhej strane tkvie nevýhoda, že ak užívateľ stroja, na ktorom herný server beží nemá záujem o účasť v hre samotnej, sú zbytočne zaberané systémové zdroje zodpovedné za grafický výstup aj keď nie je potrebný.

Druhou možnosťou je implementovať server ako samostatný program, ktorý je oddelený od grafického užívateľského rozhrania, a v prípade potreby efektívne nakladá s požadovanými zdrojmi. Jedinou nevýhodou tohto riešenia je nutnosť spustiť dva programy, na rozdiel od jedného pri predchádzajúcom. V praxi ide o majoritne používané riešenie keďže na jednom pc, ktorý zohráva úlohu servera v rámci siete, beží zväčša mnoho podobných programov.

V prípade hry Hon na ponorku je použitá druhá možnosť, z vyššie uvedených dôvodov šetrenia systémových zdrojov a budúceho vývoja, keďže pri masívnejšom nasadení a rozšírení sú na internete používané výhradne dedikované servery ako samostatné programy.

Čo sa týka funkčnosti servera, ktorú zabezpečuje, v dnešnej dobe sa v praxi používa len jediný model. Konkrétne implementácie, ako spomínaný Source engine server alebo Quake 3 engine server, sa principiálne líšia len v detailoch. Základné požiadavky, ktoré spĺňa sú:

- Obsluha viacerých spojení súčasne
- Redukcia vznikajúceho lagu
- Buffrované spracovanie
- Autoritatívna korekcia aproximovaných výpočtov
- Obsluha a detekcia kolízií

3.3.1 Výsledná špecifikácia servera

Server je navrhovaný ako konzolová aplikácia pre pripojenie maximálne súčasne ôsmich klientov, reálne je však z dôvodu lepšej hrateľnosti maximálny počet hráčov obmedzený na štyroch. Na zvolenom TCP porte 27015 očakáva prichádzajúce spojenia. V prípade úspešného pripojenia sa zo servera pošlú v textovej podobe dáta označujúce stav prihlásenia a číslo portu, na ktorom bude prijímať herné UDP datagramy. Následne sa TCP spojenie ukončí a server komunikuje s klientom výhradne pomocou UDP. Základom je nekonečná slučka, v ktorej je využívaný I/O multiplexing pre prichádzajúce požiadavky a ich následné spracovanie. Server prijíma hlavne UDP packety o stave a zmene jednotlivých hráčov. Pomocou nich simuluje dianie na mape a odosiela klientom výsledky o stave a pozícii všetkých objektov. Pre čo najlepšiu hrateľnosť sa využíva technika predikcie vstupu a interpolácie objektov. V prípade odpojenia klienta sa zas naviaže TCP spojenie so serverom a zašle sa príslušná správa, po ktorej server prestane na daný UDP port zasielať dáta. Z princípu vyplýva, že server využíva pre svoju funkčnosť viaceré systémové vlákna. Jedno pre každého klienta, plus jedno pre príjem TCP a niekoľko ďalších pre mechanizmy herných výpočtov. Výsledným efektom je sieťový program, server, ktorý dokáže spracovávať dáta efektívne a

pritom stále naslúcha spojeniam na zvolenom porte. Finálny návrh je založený na princípe činnosti sieťového servera, ktorý je súčasťou Valve Source enginu, podrobne popísanom na *developer.valvesoftware.com* [2].

3.4 Návrh klienta

Klient, ktorý je súčasťou vytvoreného herného systému architektúry klient-server, zabezpečuje prioritne základné výpočty herných mechanizmov, komunikáciu so serverom a interakciu s užívateľom.

3.4.1 Výber grafickej knižnice

Úlohou klienta je predovšetkým komunikovať s hráčom a graficky mu prezentovať aktuálne dáta v adekvátnej podobe. Preto je nutné zvoliť grafickú knižnicu, ktorá bude použitá pre tvorbu grafickej časti projektu. Po preskúmaní možností som došiel k názoru, že v súčasnej dobe je možné použiť dve rovnocenné riešenia, pričom každé ma svoje výhody aj nevýhody. Konkrétne sa jedná o DirectX alebo OpenGL.

OpenGL

OpenGL (Open Graphics Library) je priemyselný štandard špecifikujúci viacplatformové rozhranie (API) k akcelerovaným grafickým kartám, respektíve celým grafickým subsystémom. Slúži na tvorbu aplikácií pracujúcich predovšetkým s trojrozmernou počítačovou grafikou prekresľovanou v reálnom čase. Používa sa pri tvorbe počítačových hier, CAD programov, aplikácií virtuálnej reality alebo pre vedecko-technické vizualizácie. Medzi jeho výhody patrí predovšetkým multiplatformnosť. Základné vlastnosti sú štruktúrovaný prístup, podpora platforiem Windows, Linux a Mac. V porovnaní s DirectX (Direct3D) má vo väčšine prípadov výrazne kratší implementačný kód.

Direct3D

Direct3D je časť aplikačného rozhrania DirectX pre vývoj grafických prostredí od spoločnosti Microsoft. Je k dispozícii výhradne pre operačné systémy Windows a to od verzie 95 a vyššie, z čoho pramení jeho hlavná nevýhoda. Je taktiež základom grafického rozhrania hracích konzol Xbox a Xbox 360. Používa sa na renderovanie trojrozmernej grafiky v aplikáciách náročných na výkon, hlavne v počítačových hrách. Hlavnými charakteristikami sú: výhradne podpora operačných systémov Windows, objektový prístup, dlhý implementačný kód.

Zhodnotenie

Jednoznačne rozhodnúť, ktorá grafická knižnica je lepšia pre účely tohto projektu, nie je možné. Obe riešenia v podstate zabezpečujú rovnakú činnosť, poskytujú hardvérovú akceleráciu a sú výkonnostne na veľmi podobnej úrovni, tak že je iba na tvorcoch, aký prístup im viac vyhovuje. V prípade projektu sieťová hra Hon na ponorku som zvolil grafickú knižnicu DirectX (Direct3D). Dôvody môjho rozhodnutia sú nasledovné. Hra má podporovať výhradne platformu Microsoft Windows a prechod na iné platformy nie je v dlhodobom horizonte plánovaný. Využívaný objektový prístup lepšie zapadá do konceptu tohto projektu.

V súčasnej dobe sa hry v komerčnej sfére pre Windows programujú zväčša pod týmto prostredím z dôvodu zjavného dlhodobého vplyvu Microsoftu na obe riešenia. Finálny vplyv rozhodnutia, ktorú grafickú knižnicu použiť, je na celkový výsledok projektu zanedbateľný, keďže ani v prípade značného zlepšenia grafiky by sa nedospelo k bodu, kde kvality a možnosti jedného riešenia výrazne prevyšujú to druhé.

3.4.2 Grafické užívateľské rozhranie

Po úspešnom zvolení grafickej knižnice je podstatnou časťou návrhu grafické užívateľské rozhranie, pretože každý program, ktorý prezentuje dáta užívateľovi v grafickej forme, musí zabezpečiť intuitívne a ľahké porozumenie, čo dané informácie znamenajú. Tvorenie GUI výhradne pomocou DirectX je však zdĺhavé a neefektívne, preto je dobré využiť jednu z mnohých existujúcich knižníc, slúžiacich výhradne k tomuto účelu. Možnosti sú AntTweakBar, CEGUI, MyGUI a mnoho ďalších.

Z veľkého počtu riešení som zvolil najprv AntTweakBar z dôvodu jednoduchosti a ľahkej implementácie, čo sa neskôr ukázalo ako chyba, pretože možnosti knižnice nedostačovali požiadavkám. Následný prechod na CEGUI bol síce problematický, ale vyplatil sa. Výhody CEGUI spočívajú najmä v jeho komplexnosti a širokej podpore rôznych technológii od tvorby GUI pomocou XML až po LUA skriptovanie, ktoré však v tomto projekte nie je využité. Výrazným kladom riešenia je dobrá dokumentácia a fórum, kde tvorcovia knižnice radia v prípade akýchkoľvek problémov. Nevýhodami sú väčšia náročnosť na pochopenie a implementáciu, nutnosť pribalenia viacerých knižníc súvisiacich s prácou knižnice.

Po úspešnom vybratí knižnice pre tvorbu GUI a preštudovaní časti knihy *Game Development with Unity* [8] zaoberajúcou sa návrhom a spracovaním užívateľských rozhraní v hrách, som navrhol jednoduché menu v štýle hry World Of Warcraft a následne užívateľské rozhranie v hre samotnej, popísané na obrázkoch číslo 2.2 a 2.3.

3.4.3 Zobrazenie dát a korekcia so sieťou

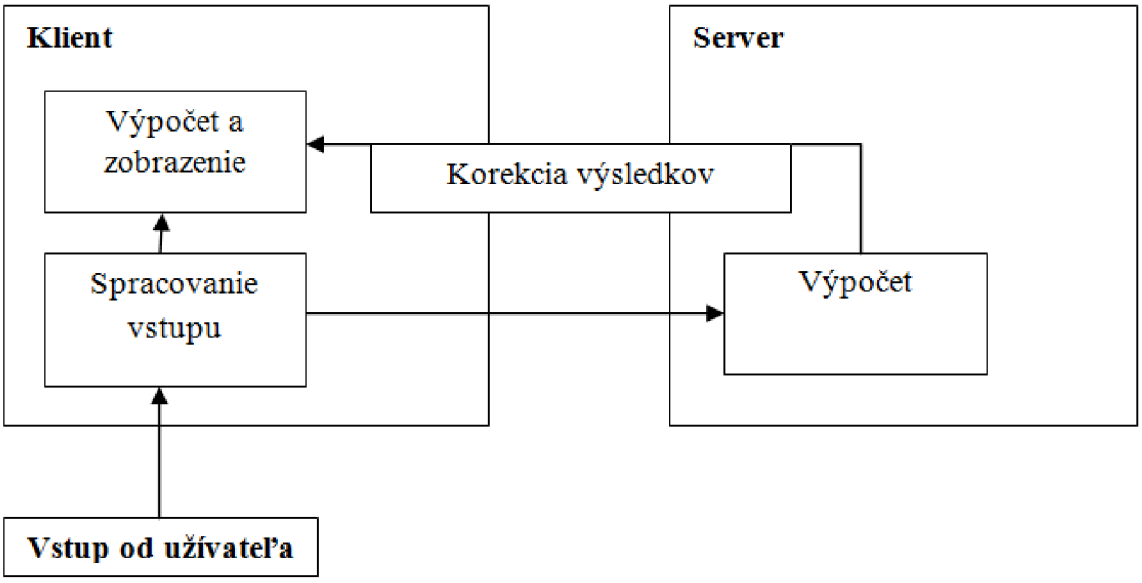
V časti návrhu sieťového protokolu je popísaný princíp jeho funkčnosti, z čoho vyplýva, že dáta budú vymieňané medzi účastníkmi hry v určitých intervaloch z dôvodu redukcie záťaže na sieti. Diskutujme nasledujúci príklad. Dáta, napríklad o pozícii objektov, budú posielané, každý konštantný časový interval 50 ms. To znamená, že pozícia objektov sa bude na strane klienta tiež meniť každých 50 milisekúnd. Z uvedených informácii je možné vypočítať, že objekty budú na obrazovke prekreslované 20 krát za sekundu a ich pozícia bude ovplyvnená časovým zdržaním na sieti, z čoho pramení veľmi neprirodzený grafický pohybový efekt v prípade hráčom kontrolovaných objektov. Z vyššie uvedého dôvodu je potrebné navrhnúť mechanizmus, ktorý to bude mať za úlohu vyriešiť. Nazvime ho predikcia vstupu.

Predikcia vstupu

Predpokladajme, že hráč má sieťovú odozvu 150 milisekúnd. Začne sa pohybovať dopredu. Informácia, že bola stlačená šípka dopredu, je spracovaná a poslaná na server, ktorý ju spracuje kódom pre korekciu aktuálnej polohy objektu a aktuálnu polohu v hernom svete zmení na vypočítaný výsledok. Táto zmena sa následne zašle klientom, takže hráč, ktorý zmenu previedol, uvidí svoju vlastnú akciu až za 150 milisekúnd. Tieto oneskorenia vplývajú na všetky akcie, ktoré hráč spraví, ako už spomínaný pohyb, streľba a pod. Čím dlhšia odozva, tým väčší diskomfort pri hre.

Odozva medzi vstupom (prevedenie akcie) a korešpondujúcim vizuálnym efektom vytvára zvláštny, neprirodzený pocit a robí aj samotný pohyb náročným, nehovoriac o streľbe alebo mierení. Predikcia vstupu na strane klienta je riešením problému. Odstraňuje odozvu a akcie sú tak zrazu oveľa prirodzenejšie. Namiesto čakania na príchod o našom aktuálnom stave zo servera, lokálny klient sám vypočítava aktuálnu pozíciu presne tým istým kódom a za použitia tých istých pravidiel ako server. Po skončení daného výpočtu sa hneď zmení aktuálna pozícia. Server v tom čase však stále vidí daný objekt na pôvodnom mieste.

Po 150 milisekundách klient prijme aktuálny obraz sveta obsahujúci zmeny, založené na predikcii vstupu klientom vypočítaných už skôr, následne ich porovná. Ak sú rozdielne, generuje sa chyba predikcie a aktuálna poloha na strane klienta sa upraví, čo je spôsobené samotnou klient-server architektúrou, keďže server je finálnou autoritou a koriguje činnosť klienta. Chyba predikcie môže byť často krát znateľná a spôsobí okamžitú zmenu pohľadu. Názorný grafický popis je zobrazený na obrázku 3.2.



Obrázek 3.2: Princíp predikcie vstupu. Obrázok znázorňuje princíp funkčnosti implementovanej techniky predikcie vstupu

Predpovedať správanie objektu je však možné len vtedy, ak klient používa presne ten istý algoritmus výpočtu pozície ako server a má informáciu o jeho budúcom smere pohybu. To väčšinou nie je práve bežný jav, pretože server pozná všetky dôležité informácie, kdežto klient len akcie daného hráča. Preto predikcia vstupu funguje iba na vlastného hráča a na objekty ním kontrolované. Predikcia vstupu ostatných hráčov alebo iných interaktívnych objektov nie je v tomto prípade možná na strane klienta. Podrobnejšie informácie je možné nájsť na *developer.valvesoftware.com* [2].

Scéna

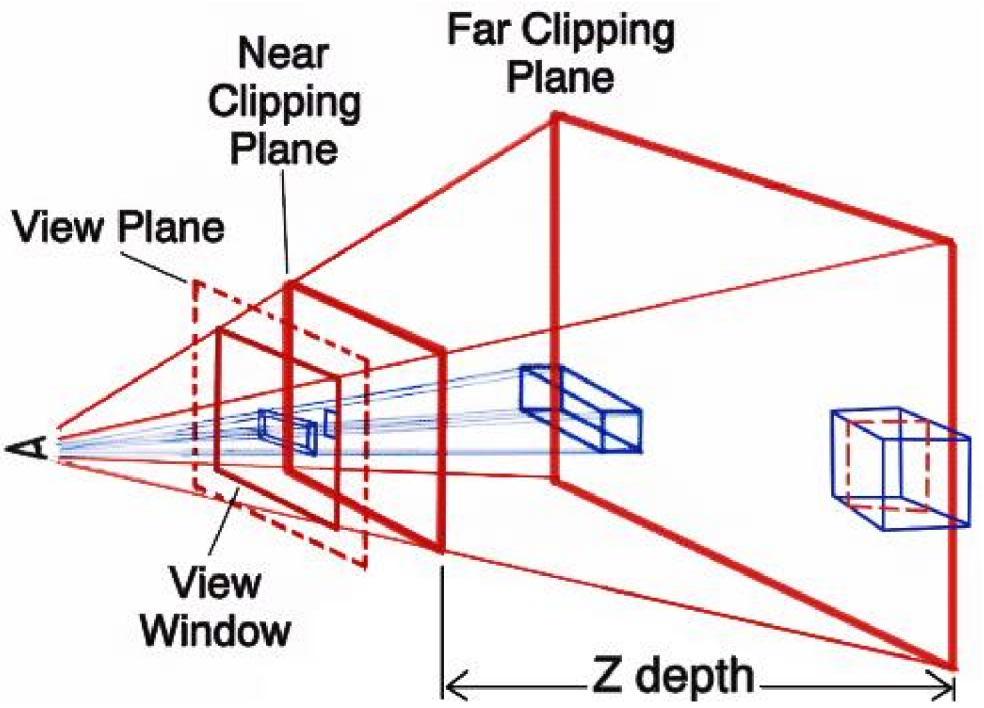
Hra je špecifikovaná nutnosťou dvojrozmernej grafiky. V tomto projekte bude však dvojrozmerná len naoko. Celá scéna je navrhnutá v trojrozmernom prostredí a výsledný 2D efekt je dosiahnutý statickou kamerou a nepohyblivou osou z. Na prvý pohľad sa to nemusí javiť ako najšťastnejšie riešenie, no na druhej strane predstavuje veľmi perspektívny potenciál

pre ďalší vývoj projektu. Výhodou tohto riešenia je, že v prípade prechodu na trojrozmernú grafiku stačí iba implementovať správny pohyb kamery a spracúvanie užívateľského vstupu pre tretiu súradnicu.

Detekcia objektov mimo scény

Predstavme si situáciu, keď hráč vystrelí raketu na ponorku. Projektil cieľ trafí, alebo ho minie. V prípade, že ho minie, jeho súradnice sú vypočítavané do nekonečna, čo môže spôsobiť mnohé problémy, ako napríklad pretečenie dátového typu a následnú možnú nežiadúcu kolíziu, poprípade iné nepredvídateľné správanie. Tomuto nežiadúcemu javu zabraňuje detekcia objektov mimo scény, využitím techniky známej pod anglickým názvom „frustum culling“. Keďže celá scéna je navrhnutá pre použitie v trojrozmernom prostredí a 2D dojem je spôsobený umelo, je pri detekcii nutné rátať s tromi rozmermi. Výhody sú opäť zrejmé pri budúcom vývoji.

Princíp metódy spočíva vo vypočítaní obrazového zorného poľa podľa matice označujúcej počiatočné súradnice zobrazenia a projekčnej matice, označujúcej koniec zorného poľa. Názorný príklad je zobrazený na obrázku č. 3.3.



Obrázek 3.3: „Frustum culling“prevzaté z [9]. Obrázok znázorňuje použitý pohľadový objem.

Pre účinnú detekciu je nutné stanoviť typ použitého výpočtu na zjednodušenie tvaru. Skúmanie, či sa všetky body aktuálneho objektu nachádzajú v zornej oblasti, by bolo zbytočne náročné na výkon, a preto je nutné použiť prístup zjednodušenia, ktorý je následne využitý aj pri samotnej detekcii kolízií objektov. Dosiahnutie dokonalého detekovania kolízií je výpočtovo aj algoritmicky veľmi náročné. V tomto projekte je využitý algoritmus vypočítania najmenšej možnej gule, obsahujúcej všetky body daného objektu. Takto celý kolízny systém pracuje len s jednoduchými tvarmi a výpočty sú rádovo jednoduchšie. Je

diskutabilné, či nie je pre tento prípad lepšie použiť princíp najmenšieho kvádra. Prax však dokázala, že pri zložitejších modeloch, ktoré by mohli byť eventuálne v budúcnosti použité, je vhodnejší guľový prístup.

3.4.4 Spracovanie užívateľského vstupu

Spracovanie údajov smerujúcich smerom od užívateľa hre Hon na ponorku je jedných z ďalších problémov, ktoré je nutné v rámci návrhu prediskutovať. V prípade tohto projektu existujú tri možnosti, ako zabezpečiť správne reakcie programu na vstupy užívateľov. Jedna z nich je využiť vývojový balík pre hry DirectX, obsahujúci okrem spomínanej grafickej knižnice Direct3D aj časť slúžiacu pre správu užívateľského vstupu s názvom DirectInput. Druhou možnosťou je využiť inú časť balíka DirectX slúžiacu pre prácu s užívateľským vstupom s názvom XInput. Poslednou možnosťou je priame spracovanie Windows správ asociovaných k vytvorenému oknu.

Rozdiely medzi DirectInput a XInput sú v prípade využitia v tomto projekte malé. XInput je novšou technológiou spoločnosti Microsoft vyvinutou hlavne z dôvodu rozšírenia programovateľnej funkčnosti zariadení ovládačov XBoxu 360, napríklad o vibrácie. Charakteristické je svojim jednoduchším použitím oproti DirectInput. Celkovo sú obe knižnice využívané najmä pri záujme o podporu alternatívnych zariadení ako joystick alebo volant.

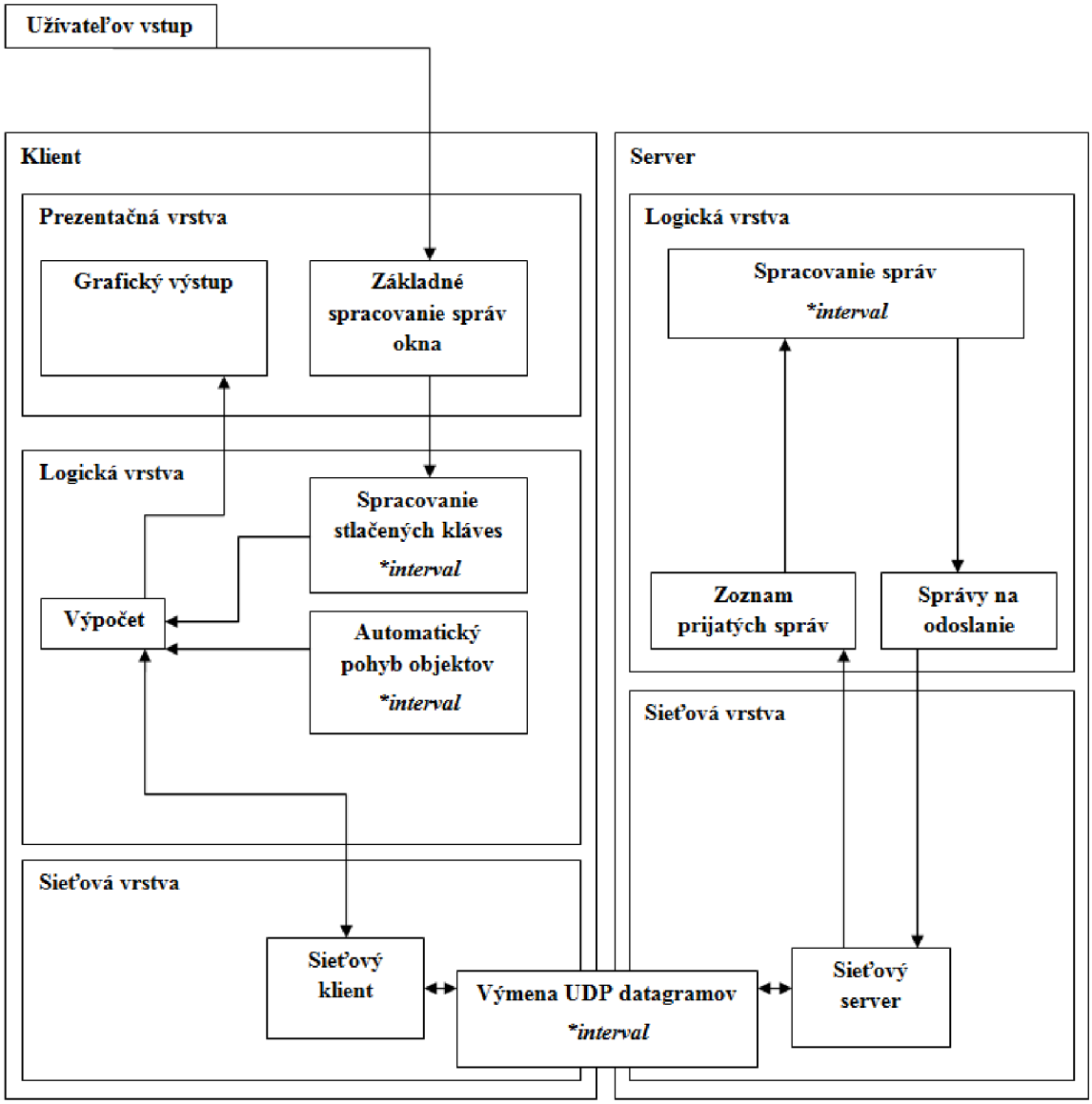
V pripade porovnania oboch riešení s jednoduchým spracovávaním Windows správ existuje jeden zásadný rozdiel. DirectInput a XInput sú nadstavbami nad existujúcimi Windows správami, takže v prípade využitia jedného z riešení DirectX balíku je kód zbytočne väčší o ďalšiu knižnicu. Z toho vyplýva, že ak aplikácia nevyžaduje podporu rozšíreného balíka zariadení a programátor si vystačí s myšou a klávesnicou, čo je aj prípad tohto projektu, je vhodné využiť spracovanie vstupu pomocou Windows správ. Podrobnejšie informácie ohľadom spracovania vstupu je možné získať z knihy *Game Coding Complete* [7] alebo mnohých ďalších kníh, dôkladne sa zaoberajúcich tvorbou hier.

3.4.5 Výsledná špecifikácia klienta

Klient je samostatný program používaný koncovými užívateľmi a zohľadňuje všetky požiadavky kladené na 2D hru. Poskytuje dvojrozmerný vizuálny efekt, založený na trojrozmernom prostredí, kompletné grafické užívateľské rozhranie vytvorené pomocou knižnice CEGUI pre používateľa s využitím balíka knižníc DirectX. Z toho vyplývajúcou platformou pre činnosť je operačný systém Windows. Základom sieťovej komunikácie je inicializácia spojenia zo strany klienta protokolom TCP zaslaním správy o pripojení na server, spolu s označením portu, na ktorom očakáva prichádzajúce UDP packety. Po obdržaní úspešnej odpovede začne odosielať správy, zohľadňujúce stav hry protokolom UDP. Klient zabezpečuje hlavne obsluhu vstupu od užívateľa, a to konkrétne zmenu pohybu, vystrelenie zo zbrane a rôzne iné činnosti. Ako už bolo spomínané, pre samotné zasielanie správ je vhodné použiť protokol UDP, pre ich veľký objem a malú veľkosť. Pomer správ posielaných protokolmi TCP a UDP je tak značne v prospech UDP. Výhody protokolu UDP sú v tomto prípade zrejmé, čo v konečnom dôsledku znamená podstatne rýchlejšiu odozvu servera ako protokol TCP.

Výsledný program, sieťový klient, je grafická aplikácia určená pre OS Windows, poskytujúca interakciu s užívateľom a v neposlednom rade zábavu. Z princípu fungovania klienta a automatického prideľovania portov je možné spustiť viac inštancií klienta na jednom systéme.

3.5 Výsledný návrh súčinnosti klienta a servera



Obrázek 3.4: Architektúra herného systému

Obrázok č. 3.4 demonštruje funkčnosť systému klient-server. Zohľadňuje použité časovania akcií a vzťahy medzi jednotlivými logickými blokmi. V prípade označenia „*interval“sa jedná o informáciu, že daná činnosť prebieha v konštantných intervaloch, stanovených na základe testovania.

Kapitola 4

Implementácia

Hlavným parametrom pri výbere programovacieho jazyka bola úroveň jeho abstraktnosti v kombinácii s požiadavkou maximálnej rýchlosti. Týmto podmienkam dokonale vyhovuje multiplatformový jazyk C++, ktorý je momentálne široko využívaný naprieč celým herným odvetvím. Dovoľuje využitie objektovo orientovaného programovania a z toho vyplývajúceho množstva výhod. Treba však podotknúť, že jazyk C++ nie je čisto objektovo orientovaný, ale umožňuje akúsi kombináciu štýlov, čo je opäť možné brať ako nezanedbateľnú výhodu.

4.1 Voľba vývojového prostredia

Pri vybratom programovacom jazyku a štýle, treba už len vhodne zvoliť vývojové prostredie. Keďže plánovaný vývoj je len pre platformu Windows, jasnou voľbou je komerčný nástroj od spoločnosti Microsoft s názvom Visual Studio 2010. Ten poskytuje dnes momentálne najlepší možný debugging, podporu, manažment projektu a priateľskosť užívateľského rozhrania spomedzi všetkých konkurenčných vývojových prostredí. Študentskú licenciu vo verzii Professional je zadarmo možné získať pre nekomerčné použitie prostredníctvom programu Microsoftu na podporu študentov s názvom MSDNAA.

4.1.1 Štruktúra projektu

Zvolené vývojové prostredie Microsoft Visual Studio 2010 disponuje veľmi kvalitným manažmentom riešení. Pre potreby sieťovej hry Hon na ponorku je vytvorené jedno riešenie (solution) obsahujúce dva samostatné projekty - klient a server. Pre úspešné preloženie je potrebný vývojový balíček DirectX 9 (DirectX SDK) a nastavenie príslušných ciest v správe projektu klient. Prezentácia dát výsledným programom závisí od zvolenej konfigurácie. V prípade ak ide o „debug“verziu, je na klientovi, okrem grafického okna, zobrazená aj konzola pre textový výstup. V prípade „release“verzie sa všetok textový výstup ukladá do súboru v aktuálnom adresári.

4.2 Implementácia sieťového protokolu

Pre správne fungovanie sieťovej aplikácie je nutné efektívne implementovať sieťový protokol. Nasledujúca kapitola popisuje vytvorenie virtuálneho spojenia nad protokolom UDP, tvorbu hlavičiek, ich obsah a vysvetľuje nutnosť a princíp ich použitia. Popísané sú základné triedy

Tabulka 4.1: Použité názvoslovie v kóde

Konštrukcia	prefix
Trieda (class)	SMC
Štruktúra (struct)	SMS
Rozhranie (interface)	SMI
Členská premenná	m_
Globálna premenná	g_

pre prácu s protokolom, ich funkčnosť a začlenenie do celku. Štruktúra sieťovej správy bola postavená na princípe informácií získaných z *Glenn Fiedler's Game Development Articles and Tutorials* [5].

4.2.1 Virtuálne spojenie

Teraz späť k sieťovým protokolom. Pri práci s TCP socketmi je na prvý pohľad zrejmé, že je vytvorené akési spojenie. TCP je implementované nad IP vrstvou a IP sú predsa len packety prenášané z jedného počítača do druhého. Z toho vyplýva, že tento koncept spojenia musí zapuzdrovať len akési virtuálne spojenie. No a ak TCP môže vytvoriť virtuálny koncept spojenia nad IP, prečo by nebolo možné niečo podobné spraviť aj nad UDP? Definujme virtuálne spojenie ako dva počítače, vymieňajúce si navzájom UDP packety fixnou frekvenciou. Ak packety budú posielané a prijímané, teda bude existovať dátová prevádzka, považujme tieto dva počítače za virtuálne spojené.

4.2.2 Protokol id

Keďže UDP je založené na koncepte nevytvárania spojenia, UDP socket dokáže prijať packety poslané z ktoréhokoľvek počítača v rámci siete. V našom prípade je potrebné určiť limity a obmedziť prijímanie správ natoľko, aby klient akceptoval iba správy poslané zo servera. A naopak, server akceptoval len správy poslané z klienta. Nemôžeme filtrovať packety len na základe adresy, keďže ich server dopredu nepozná. Namiesto toho použijeme ako prefix každého UDP datagramu malú hlavičku, ktorá obsahuje 32 bitový identifikátor protokolu. Protokol id je však len obyčajné, náhodné, raz stanovené číslo, reprezentujúce vytvorený protokol. Každý packet, ktorý dorazí, je podrobený testu, či jeho hlavička obsahuje stanovené id. Ak kontrola prvých štyroch bytov nepreukáže zhodu identifikátorov, packet je ignorovaný. Ak pri porovnaní nastane zhoda, hlavička sa oreže a následne dáta sa spracujú.

4.2.3 Sekvenčné číslo

Cieľ výsledného systému správ, implementovaného nad UDP, je jednoduchý - dosiahnuť spoľahlivý prenos dát. Pri riešení tohto problému je najskôr nutné identifikovať packety. Na to slúži pravé sekvenčné číslo. Každý packet, ktorý je poslaný, má svoje unikátne sekvenčné číslo. Nasledujúci packet má číslo zväčšené o jeden. Toto je jedna z bežných techník, ktorú dokonca využíva aj TCP. Cieľom však nie je, a ani nikdy nebolo, implementovať spoľahlivé doručenie správ presne tak, ako to robí TCP.

UDP však negarantuje poradie prenosu packetov, takže stý prijatý packet nemusí byť zároveň stým packetom odoslaným. Tým pádom je nutné rozšíriť hlavičku o sekvenčné číslo.

V prípade, kedy je packet so sekvenčným číslom novší ako posledný prijatý packet, spracuje sa. V opačnom prípade je packet zahodený.

4.2.4 Potvrdenie o prijatí správ

Okamžite potom, čo vieme identifikovať protokol a určiť poradie packetu je ďalším krokom doručenie a spracovanie potvrdení o prijatých packetoch. Logicky je to celkom jednoduché. Je potrebné si len zapamätať jednotlivé sekvenčné čísla prijatých packetov a následne ich zaslať spať. Pretože vytvorené spojenie zasiela packety s určitou frekvenciou, stačí pridať ďalšiu položku do hlavičky a to 32 bitové číslo, nesúce informáciu sekvenčného čísla potvrdeného packetu.

Principiálny prístup je nasledovný. Každý krát čo je zaslaný packet, sa zvýši lokálne sekvenčné číslo. Pri prijatí packetu sa dané sekvenčné číslo packetu skontroluje a porovná sa s doterajším najvyšším prijatým číslom packetu, nazývaným vzdialené sekvenčné číslo. Ak je packet novší, tzn. obsahuje novšie sekvenčné číslo ako doteraz posledný prijatý packet, upravíme spomínané číslo na číslo packetu. Pri skladaní hlavičky packetu sa lokálne sekvenčné čísla stávajú sekvenčné čisla packetu a vzdialené číslo sa stáva potvrdením o prijatej správe.

Tento jednoduchý systém potvrdzovania správ umožňuje v každom poslanom packete potvrdiť packet prijatý. Ale čo ak pár packetov dorazí skôr, ako sme schopní sami poslať packet? V hlavičke je totiž miesto len pre jedno potvrdenie. Zvážme prípad, ak jedna strana posiela packety inou frekvenciou ako strana druhá. Naskytá sa problém, čo s tým. Tu je ďalší rozdiel oproti TCP.

4.2.5 Spoľahlivý prenos potvrdení o prijatých správach

TCP v základe využíva algoritmus sliding window, čo je presne správanie, ktoré nie je požadované. V projekte je implementovaný systém prenosu packetov, v ktorom sa nikdy neposiela packet s rovnakým sekvenčným číslom viac ako raz. Každý ďalší packet má číslo zvýšené o n+1, n+2 a tak ďalej. Nikdy, naozaj nikdy sa systém nezastaví a stratený packet sa zašle znovu. V prípade, ak stratené dáta boli dôležité pre chod aplikácie, musia byť poslané v novom packete s novým sekvenčným číslom. Je zbytočné totiž opakovane zasielať packety, ktoré obsahovali málo potrebné alebo neaktuálne informácie.

Vyššie uvedený rozdiel s TCP spôsobuje možné vytváranie dier v potvrdeniach. Preto nie je dostačujúce v každom packete posielať iba jedno potvrdenie o prijatí. Je nutné ich tam pridať niekoľkonásobne viac.

Ako bolo spomenuté skôr, povedzme, že jedna strana posiela packety inou frekvenciou ako druhá. Predpokladajme, že v najhoršom prípade to bude nie menej ako 10 packetov za sekundu, pričom druhá strana nie viac ako 30. V tomto prípade, priemerný počet potvrdení, ktoré je nutné poslať v jednom packete je 3, ale ak sa situácia zhorší a packety sa nakopia, bude ich treba viac, povedzme 6-10 v krajnom prípade. Ale čo potvrdenia, ktoré neprišli z dôvodu straty packetu?

Na vyriešenie tohto problému je použitá klasická stratégia redundancie viacerých potvrdení. Do každého packetu je vložených vždy presne 33 potvrdení predchádzajúcich packetov. Toto magické číslo je vypočítané z veľkosti 32 bitového integera, plus jeden z dôvodu prvého identifikačného potvrdenia. Využité je totiž bitové pole. Trik spočíva v nastavení bitov podľa počiatočného potvrdenia. Napríklad integer potvrdenie v hlavičke obsahuje posledné číslo prijatého packetu a v bitovom poli sú bity nastavené tak, aby určovali doručenie/stratu po-

sledných 32 packetov od základného potvrdenia. Tým je dosiahnutá obrovská redundancia potvrdení za veľmi malú cenu.

Vyššie uvedené skutočnosti sú v kóde implementované pomocou triedy SMCPacket, zaoberajúcou sa serializáciou hlavičky, pozostávajúcej z popísaných prvkov, plus aplikačných dát, deserializáciou, nastavovaním jednotlivých potvrdení a bitového poľa.

4.2.6 Serializácia a deserializácia

Dáta sa počas behu programu nachádzajú v pamäti počítača a nie je možné ich bez úprav posielať. Pre prenos po sieti je nutné vytvoriť vhodnú formu štruktúrovaných dát ako súvislé poradie bytov s presne stanovenou veľkosťou. Tento proces sa nazýva serializácia. Štruktúrované dáta možno do streamu zapísať viacerými spôsobmi. Základné rozdelenie formátov je na binárne a textové. Použitie oboch spôsobov sa spája s určitými problémami, ktoré je potrebné vyriešiť.

- V binárnom súbore sú zapísané bajty dát, tak ako sú reprezentované v pamäti počítača. Pre každú položku štruktúry je v súbore toľko bajtov, koľko ich zaberá v pamäti jeho dátový typ.
- V textovom súbore sú dáta uložené v takej forme, v akej sú obvykle prezentované používateľovi v používateľskom rozhraní. Dáta v textovom súbore sú pre človeka jednoducho čitateľné.

V tomto projekte je využitá kombinácia oboch princípov, v prevažnej miere sa však používa serializácia binárna a to hlavne pri samotnom posielaní herných správ nad UDP. Textová forma serializácie je využitá pri posielaní správ o prihlásení a odhlásení nad TCP.

Deserializáciou sa rozumie proces vytvorenia štruktúry zo sieťovej podoby.

4.2.7 Trieda SMCPacket

Forma serializovaných dát v triede SMCPacket:

1. Číslo protokolu - protokol id
2. Sekvenčné číslo
3. Potvrdenie o prijatom packete
4. Bitové pole
5. Ostatné aplikačné dáta (SMCNetworkMsg)

V aplikačných dátach trieda SMCPacket zapúzdruje triedu SMCNetworkMsg.

4.2.8 Trieda SMCNetworkMsg

Trieda SMCNetworkMsg obsahuje aplikačné dáta, ktoré pozostávajú zo zoznamu (list) tried typu SMCGameMsg, typ správy a počtu správ v zozname. Zapúzdruje akýsi medzistupeň medzi najnižšou UDP vrstvou a aplikačnými dátami, ktoré sú neskôr predostreté samotnej logike aplikácie. Trieda implementuje hlavne serializáciu zoznamu. Ak zoznam nie je prázdny, typ správy je MSG_DATA. Ak je prázdny, nastaví sa typ správy na MSG_OK. V

tomto prípade správa slúži len na udržanie vytvoreného virtuálneho spojenia, aby nedochádzalo k nevyžiadanému timeoutu na oboch stranách.

Forma serializovaných dát v triede SMCNetworkMsg

1. Lineárny zoznam štruktúr SMCGameMsg
2. Typ správy (MSG_OK/MSG_DATA)
3. Počet prvkov v zozname

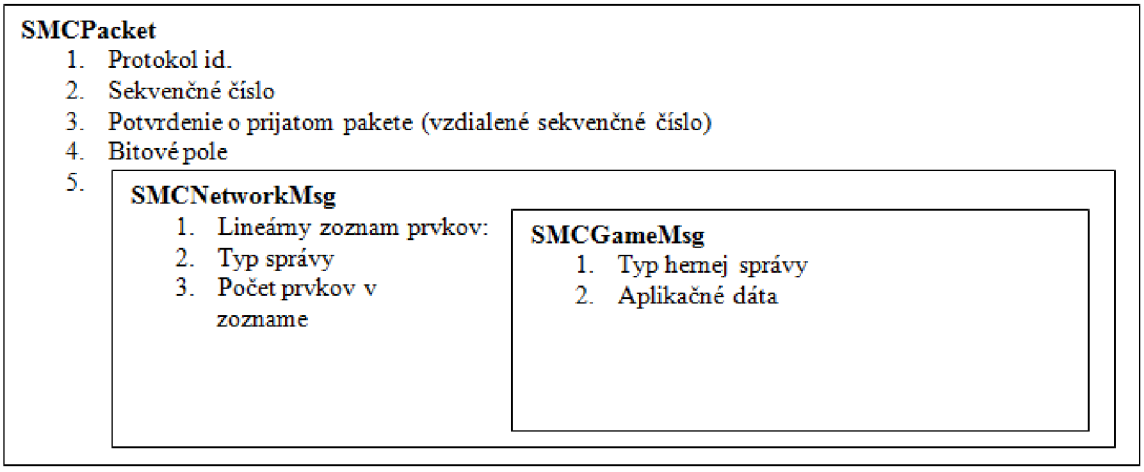
4.2.9 Trieda SMCGameMsg

Trieda SMCGameMsg obsahuje už samotné aplikačné dáta, vytvorené a spracovávané logikou aplikácie. Pozostáva z typu správy a ostatných dát, ktorých obsah je tvorený v závislosti na posielanom stave.

Forma serializovaných dát v triede SMCGameMsg

1. Typ hernej správy
2. Dáta

4.2.10 Výsledná štruktúra hlavičky a dát



Obrázek 4.1: Štruktúra správy. Na obrázku je znázornená štruktúra správy packetu.

4.3 Implementácia servera

Hra pre viac hráčov s názvom Hon na Ponorku využíva klient-server sieťovú architektúru. Server je vždy dedikovaný a beží na ňom celá logika hry. V konfrontácií s klientom je vždy, vzhľadom na výpočty, autoritatívny. Keďže sieťová priepustnosť je obmedzená, server neposiela každú vypočítanú zmenu všetkým hráčom hneď, ale v určitej frekvencii. Packetom

samozrejme trvá nejaký čas prekonať fyzickú vzdialenosť medzi dvoma počítačmi, čo znamená, že čas na klientovi je vždy trochu oneskorený oproti serveru (ping time). Packety poslané od klienta, ako napríklad užívateľský vstup, tiež majú určité oneskorenie.

Architektúra servera pozostáva z dvoch vrstiev:

- Logická vrstva
- Sieťová vrstva

4.3.1 Logická vrstva

Každá aplikácia, zabezpečujúca určitú funkčnosť, obsahuje kód, ktorý je základom pre potrebné matematické výpočty, čo sa bežne označuje ako logika aplikácie. Súčasťou tejto vrstvy sú algoritmy pre výpočet nových pozícií, kolízií, spracovania herných správ a podobne.

Spracovanie herných správ

Pri efektívnom a výpočtovo úspornom spracovaní vstupu je nutné dbať, aby nebol spracovaný v okamihu príchodu, ale pri určitej frekvencii. Server simuluje hru v diskrétnych časových krokoch, nazývaných tik. Základné nastavenie je časový krok nastavený na hodnotu 20 ms tak, že za sekundu sa svet simuluje presne 50 krát. Pri výpočte každej novej simulácie, server spracuje užívateľský vstup, vypočíta kolízie, skontroluje pravidlá hry a obnoví stav všetkých objektov. Vyššia frekvencia tikov zvyšuje presnosť simulácie za cenu zvýšeného využitia procesoru a taktiež vyžaduje väčšiu sieťovú priepustnosť na oboch stranách.

- Server spracúva buffrovaný užívateľský vstup v pravidelnej frekvencii
- Server následne zasiela obraz aktuálneho sveta v pravidelnej frekvencii

Detekcia kolízií

Principiálne každá počítačová hra vyvinutá v dnešnej dobe zabezpečuje aspoň základný výpočet kolízií a analyzuje vzájomnú polohu objektov. V prípade sieťovej hry: Hon na ponorku je algoritmus časťou modul serverového programu. Pri každej simulácii nových polôh objektov v aktuálnom svete je prevedená detekcia kolízii všetkých objektov na scéne netriviálnym spôsobom. Využitý algoritmus zahŕňa zjednodušenie objektov technikou najmenšej gule, ktorý je použitý pri prvotnej detekcii kolízie. V prípade ak nastala, je pre spresnenie výsledku využitý algoritmus pre detekciu na úrovni polygonov. Výsledný princíp je znázornený nasledujúcim pseudokódom:

```
oSizeSum = objekt1.GetSphereRadius() + objekt2.GetSphereRadius();
xDistance = fabs(objekt1.X() - objekt2.X());
yDistance = fabs(objekt1.Y() - objekt2.Y());
zDistance = fabs(objekt1.Z() - objekt2.Z());

if (xDistance < oSizeSum && yDistance < oSizeSum && zDistance < oSizeSum)
{
```

```
    If(Polygonialna_kolizia());
        Return kolizia ;
}
```

Treba však podotknúť, že polygoniálny fyzikálny model sa pre zjednodušenie výpočtov líší od grafického. Táto technika sa väčšinou používa pri zložitých grafických modeloch, čo nie je prípad hry Hon na ponorku, ale pre korektnosť implementácie fyziky podľa dnešných štandardov bola implementovaná.

4.3.2 Sieťová vrstva

Sieťové rozhranie je implementované pomocou Windows socketov, konkrétne knižnice Winsock2, zapuzdrené v triede SMCNetworkServer, ktorá zabezpečuje štart, korektné ukončenie a manažment sieťových pripojení. Využíva triedy pre spracovanie komunikačného sieťového protokolu a zabezpečuje spoľahlivý prenos dát z klienta na server, posielanie a skladanie správnych potvrdení o prijatých packetoch. Pri posielaní serverových packetov sa neuplatňuje princíp spoľahlivého doručenia správ a to z nasledujúceho dôvodu. Nie je totiž žiadúce stratené packety posielať znovu a to z dôvodu neaktuálnosti. V princípe, keď sa packet stratí a eventuálne by prišla žiadosť na jeho znovu odoslanie, je zväčša mnoho objektov už v inom stave a pôvodný packet by neobsahoval aktuálne dáta. Preto je tento princíp použitý iba na strane klienta a v prípade servera sa každý jeden packet s aplikačnými dátami posiela raz. V prípade klienta sa síce tiež posiela každý packet iba raz, ale pri nedoručení sú aplikačné dáta zaslané v novom packete. Na strane servera sú zahodené. To výrazne zjednodušuje celý proces komunikácie. Problém by mohol nastať v prípade veľkého počtu objektov na scéne, kedy by sme narazili na fyzický limit UDP datagramov. To však v tomto konkrétnom prípade nehrozí.

Základnou požiadavkou na server bola nutnosť spracovávať a obsluhovať mnoho aktívnych pripojení naraz. To je dosiahnuté viacvláknovým prístupom a následnou synchronizáciou s manažmentom zdieľanej pamäti. Server čaká na prichádzajúce požiadavky v základnom vlákne na stanovenom TCP porte. V prípade úspešného spojenia od klienta, obdrží správu s loginom a číslom portu v textovej podobe. Následne spracuje danú požiadavku, zistí či klient spĺňa všetky náležitosti a v prípade úspešnej verifikácie pridá klienta do zoznamu aktívnych klientov. Následne mu zašle špecifické číslo portu, na ktorom bude on prijímať UDP dáta. Po vykonaní predchádzajúcej akcie začne v pravidelných intervaloch zasielať na danú adresu logické správy, generované serverom. V prípade, že jedna strana prestane dáta odosielať, spojenie sa preruší po uplynutí určitej doby a následne sa uvoľnia systémové zdroje, súvisiace s daným klientom. V prípade korektného ukončenia klienta server obdrží na novom TCP spojení správu o odpojení a následne uvoľní príslušné zdroje.

Príklad TCP správy, označujúcej pripojenie nového klienta:

Správa „hello“: [1][\6]20010

Prvý bajt znamená, že klient je v stave menu. Druhý bajt hovorí , že sa chce pripojiť, posiela nezobraziteľný ASCII znak. Následné bajty označujú číslo portu v textovej podobe, na ktorom klient prijíma UDP dáta.

Pri vytváraní servera boli použité pokročilé techniky programovania, ako využitie viacvrstvovej architektúry, návrhových vzorov, napr. jedináčik (singleton), ktorý je dizajnovo

vhodný najmä pre prístup k systémovým zdrojom ako GUI , zápis do súborov, atď. alebo wrapper.

4.4 Implementácia klienta

Pre implementáciu a následný preklad klienta sieťovej hry: Hon na ponorku je potrebné mať nainštalovaný DirectX9 vývojársky balíček. Táto kapitola popisuje základnú architektúru programu a jej jednotlivé časti. Následne odhaľuje, vysvetľuje dôvody a techniky, použité pri riešení jednotlivých problémov.

4.4.1 Architektúra

V časti klient je použitý koncept štandardnej programovej trojvrstvovej architektúry na strane klienta.

- Sieťová vrstva - zaoberá sa komunikáciou so sieťou
- Prezentačná vrstva - prezentuje výstup užívateľovi
- Logická vrstva - zabezpečuje výpočet a správu herných mechanizmov

Sieťová vrstva zabezpečuje prístup, komunikáciu, vytvorenie a udržiavanie spojenia, vytváranie hlavičiek, serializáciu a deserializáciu sieťových tried. Komunikuje s logickou vrstvou, ktorá spracúva prijaté správy v podobe štruktúr, analyzuje a vypočítava nové polohy objektov aktuálneho hráča, spracúva vstupy kláves a následne predostiera dáta prezentačnej vrstve, ktorá ich zobrazuje užívateľovi.

4.4.2 Sieťová vrstva

Sieťová vrstva zabezpečuje základné spojenie so serverom, zapuzdrené v triede SMCNetworkClient, ktorá vytvára jednotlivé vlákna pre príjem a posielanie UDP dát. Zabezpečuje taktiež spoľahlivý prenos aplikačných dát a manažuje celú prebiehajúcu sieťovú komunikáciu využitím tried pre prácu so sieťovým protokolom.

Spoľahlivý prenos dát z klienta na server

Pri odosielaní dát z klienta na server je vhodné zabezpečiť spoľahlivý prenos. Predstavme si, že správa s užívateľským vstupom sa stratí. Technika predikcie vstupu na strane klienta však počíta s tým, že správa bola úspešne doručená na server. V princípe nie je podstatný rozdiel pár milisekúnd, dôležité je však to, že klient sa môže spoľahnúť na úspešné doručenie. V prípade, ak by sa tak nestalo, dochádza k desynchronizácii a z toho vyplývajúcich následných problémov. Všetky odosielané packety sú uchovávané dovtedy, pokiaľ nie je doručená odpoveď zo servera o ich prijatí, alebo uplynie štandardná doba jednej sekundy, po ktorej sa packet považuje za stratený. O správnu prácu s poslanými packetmi sa stará trieda SMCPacketQueue, zabezpečujúca uchovávanie, správu a odstraňovanie packetov. V prípade uplynutia spomínanej doby jednej sekundy, sa packet zo zoznamu odstráni a aplikačné dáta, ktoré obsahoval, sú poslané v novom packete.

4.4.3 Prezentačná vrstva

Na to, aby užívateľ vedel efektívne a cieľavedome kontrolovať program, musí mu byť predostretý adekvátny grafický výstup, čo zabezpečuje prezentačná vrstva. Využité sú grafické knižnice Microsoft DirectX9, umožňujúce kompatibilitu so systémami Windows XP a novšími. O celý grafický výstup sa stará trieda SMGUI, navrhnutá pomocou návrhového vzoru jedináčik, ktorá zabezpečuje vytvorenie Windows okna, následnú inicializáciu grafických prostriedkov, vykresľovanie objektov a GUI. V prípade ukončenia programu, zabezpečí korektné uvoľnenie všetkých zdrojov týkajúcich sa grafických prvkov.

Grafické užívateľské rozhranie

Základným kameňom pri tvorbe GUI v projekte sieťová hra Hon na ponorku je komplexná grafická knižnica CEGUI. Prvky rozhrania boli namodelované v programe CELayoutEditor, ktorého výstupom je súbor XML obsahujúci jeho popis. Následne je pri inicializácii knižnice v triede SMGUI súbor načítaný a pridaná charakteristika obsluhy jednotlivých komponentov.

4.4.4 Logická vrstva

V nasledujúcej časti práce je popísaná logika aplikácie na strane klienta, jednotlivé štruktúry a postupy, použité pri riešení vyskytnutých problémov.

Objekty

Objekty tvoria základ celej scény a sú najdôležitejšou súčasťou celej hry. Pod týmto pojmom sú označené všetky torpéda, ponorky a ostatné zobrazené entity na obrazovke. Základnou programovou štruktúrou pre manipuláciou s objektmi je rozhranie SMIObject, ktoré zabezpečuje kompletnú činnosť, súvisiacu so sieťovou manipuláciou. Konkrétne rôzne serializačné, deserializačné metódy, prácu s pozíciami, verifikáciu prichádzajúcich pozícií a korekciu s technikou predikcie vstupu. Pri vytváraní nového typu objektu je nutné dané rozhranie zdediť, následne implementovať virtuálne metódy a definovať nové body, vytvárajúce finálny tvar. Pre efektívne oddelenie prezentačnej a dátovej vrstvy pre prácu s objektmi slúži trieda SMIObjectDraw, zabezpečujúca vykreslenie a správu grafickej pamäti.

Generovanie objektov

Nie všetky objekty, zobrazené na scéne, sú viditeľné od štartu aplikácie. Rôzne projektily alebo objekty podobného charakteru sú závislé na vývoji hernej situácie a musia byť generované za chodu. Pre tento účel slúži systém herných správ, informujúci oba subjekty client-server architektúry o nových vzniknutých udalostiach. Následným spracovaním sú vytvorené nové objekty, poznamenané dátami o ich rodičovskom objekte (napríklad vzťah torpédo-ponorka) a to hlavne o poslednej pohybovej akcii, z ktorej je následne možné vypočítať budúce pozície strely.

Užívateľský vstup

Kvôli prevencii voči rôznym problémom s príchodom systémových správ, označujúcich

užívateľskú aktivitu, je vstup spracovávaný v samostatnom vlákne. Každú pevnú časovú periódu je skontrolovaný zoznam práve stlačených kláves a následne sú vygenerované prislúchajúce herné správy. Použité klávesy pre pohyb, streľbu sú fixne definované, ale je implementovaná trieda SMCKeyboardMap, ktorá umožňuje ich jednoduché premapovanie pre potreby užívateľa. Program aktívne skenuje zoznam práve stlačených kláves a vykonáva príslušné akcie. V prípade detekcie akcie je k práve aktívnemu objektu (ponorke) priradený určitý typ príkazu, ktorý sa následne spravuje. V prípade ak sa jedná o pohyb, je pre ponorku vygenerovaná nová pozícia s parametrami:

- Súradnice x, y, z
- Časová známka
- Unikátny identifikátor novej pozície, v ktorom je zakódovaná príslušnosť ku klientovi

Následne, je pozícia serializovaná a poslaná na server, ako typ hernej správy, ktorý ju analyzuje a spracuje. V prípade ak je zaznamenaná udalosť „streľba“, vygeneruje sa nový objekt typu torpédo s rôznymi parametrami. Najdôležitejšie z nich sú:

- Identifikátor rodičovského objektu
- Aktuálny smer
- Rýchlosť pohybu

Po úspešnom vytvorení torpéda sa následne vypočíta nová pozícia s totožnými parametrami ako v prípade ponorky, vygenerujú sa dve herné správy a zašlú na server.

Fyzika pohybu

Pre aproximáciu reálneho správania sa telies pohybujúcich v kvapaline, je fyzika pohybu implementovaná pomocou detekcie dĺžky po sebe idúcich impulzov demonštrované nasledujúcim pseudokódom.

```
if(predchádzajúci_smer pohybu == smer_pohybu)
{
    if(násobič_pobybu < max_nasobič)
    {
        násobič_pobybu *= konštantná_zmena;
        if(násobič_pobybu > max_nasobič)
        {
            násobič_pobybu = max_nasobič;
        }

    }
}
else
{
    predchádzajúci_smer pohybu = smer_pohybu;
    násobič_pobybu = východzia_hodnota;
}
```

Výsledný aktuálny posun sa vypočíta vynásobením násobiča pohybu s maximálnou rýchlosťou pohybu, pričom maximálna hodnota násobiča môže dosiahnuť hodnotu 1, kedy je pohyb rovný definovanej rýchlosti v popise objektu.

Automatický výpočet nových pozícii

Predstavme si, že užívateľ stlačí klávesu Streľba. Celý proces začína spracovaním užívateľského vstupu, nasleduje generovanie nového objektu, zaslanie správy o aktuálnej pozícii a rôznych iných dôležitých dát na server, ktorý neskôr odpovedá a verifikuje pozície. Objekt je vytvorený, zobrazený, dokonca aj server už má o ňom informácie. Stále však nie je zabezpečená jeho základná vlastnosť a tou je automatický pohyb. Vygenerované torpédo je statický objekt, ktorý už nedostáva žiadne iné príkazy od herného mechanizmu. Riešením je automatický pohyb objektov tejto kategórie. V sieťovej hre Hon na ponorku je pre výpočet nových pozícii použitý štandardný algoritmus výpočtu pozície, implementovaný v rozhraní SMIObject, ktorého parametrami sú emulované klávesové zmeny pozície len pre ten konkrétny objekt s určitou periódou. Výsledným efektom je automatický pohyb v poslednom smere hlavného objektu - ponorky.

Navádzanie torpéd

Navádzanie torpéd v hre Hon na ponorku pozostáva v upravovaní smerovania torpéda pri automatickom výpočte novej pozície. Pri každom prepočte sú analyzované polohy všetkých ponoriek v smere strely, následne je vybraná ponorka a je vygenerovaná podobná správa ako pri stlačení klávesy pre daný objekt, ktorá je spracovaná. Vypočíta sa nová poloha torpéda v závislosti na jeho definovanej rýchlosti v smere, ktorý vypočítala metóda triedy SMCLogic GetDirectionOfClosestEnemy(). Činnosť je možné upresniť nasledujúci pseudokódom.

```
torpédo;
for(it = objekty.begin(); it < objekty.end(); it++)
{
    if((*it)->GetObjectType() == ponorka)
    {
        Získaj polohu objektu voči torpédu;
        if(torpédo smeruje na ponorku)
        {
            vypočítaj vzdialenosť od torpéda k ponorke;
            if(vzdialenosť menšia ako predchádzajúca najmenšia vzdialenosť)
            {
                najkratšia vzdialenosť = vzdialenosť;
            }
        }
    }
}
smeruj na ponorku v najkratšej vzdialenosti;
```

V prípade minutia jednej ponorky sa torpédo aktívne navádza na ďalšiu v jeho smere.

Kapitola 5

Testovanie a ladenie

Zvolené vývojové nástroje od spoločnosti Microsoft ponúkajú dobré možnosti ladenia celého riešenia. Microsoft Visual Studio debugger na mňa spravil opäť fajn dojem. Možné skoky v kóde, prepisovanie a prezentácia hodnôt premenných, zobrazenie paralelných zásobníkov jednotlivých vlákien a história volaných príkazov sú vychytávky, ktoré som pri ladení najviac ocenil.

Testovanie herného systému bolo náročnou úlohou pre mňa a viacerých ľudí na rôznych miestach a sieťach s rozdielnymi parametrami, úrovňou záťaže a kvalitou spojenia.

5.1 Testovanie sieťovej komunikácie

V aktuálnej verzii sieťovej hry Hon na ponorku nie je možné prevádzkovať sieťový server na internete ale iba na lokálnej sieti (LAN). Celý sieťový systém, vrátane protokolu, je však navrhnutý tak, aby bol použiteľný aj v reálnej internetovej prevádzke s malými nárokmi na sieťové zdroje.

Testovanie herného systému prebiehalo na rôznych sieťach s rozdielnymi parametrami straty packetov a odozvy. A to konkrétne :

- Malá domáca LAN sieť s počtom počítačov do 10 a wifi
- Stredne veľká firemná LAN sieť s počtom počítačov 100-200
- Wifi sieť v kaviarni

V prípade domácej siete bola úloha jasná, overiť funkčnosť programu v malej uzavretej skupine počítačov s parametrami káblového spojenia rýchlosti 100 Mb/s a stratou packetov do jedného percenta a odozvou 5 ms. Prítomnosť wifi typu b však umožňovala moduláciu kvality spojenia priblížením a vzdialením sa od prístupového bodu, tak že určite nebolo možné povedať, že podmienky boli ideálne. Testované boli naraz 4 aktívne pripojenia a test dopadol úspešne.

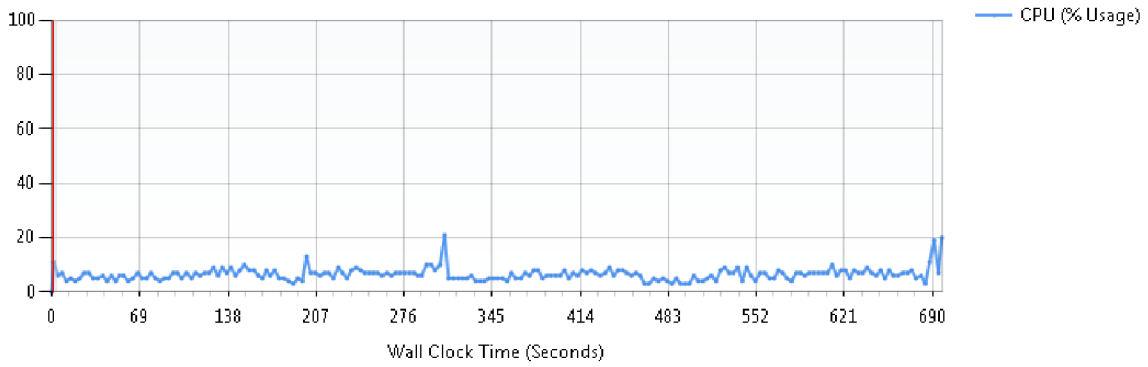
V prípade testu v stredne veľkej firme bolo účelom testovať chod aplikácie pri vyťaženej sieti v pracovnej dobe. Sieť obsahovala približne 150 počítačov, mala odozvu 5 až 50 milisekúnd a packetloss do 10 percent. Naraz sa testu zúčastnili 3 aktívni klienti a test dopadol taktiež ako v prvom prípade úspešne.

V poslednom prípade testovania v kaviarni bolo účelom zistiť, či aj pri vysokej strate packetov je hra stále hrateľná. Parametre wifi sieťe boli: typ g, tzn. 54 Mb/s, signál bol však tak slabý že packetloss dosahoval 50 a viac percent a odozva bola v niektorých prípadoch aj

nad 1000 ms. Ako bolo predpokladané, test dopadol neúspešne a hra je v takomto prostredí úplne nehrateľná.

Výsledkom činnosti testovania a ladenia bola hra odladená pre odozvy, vyskytujúce sa v lokálnej sieti bez extrémnej straty packetov a odozvy do 50 ms. Správanie a korekcia lagu pri vyšších odozvách je cieľom ďalšieho vývoja v budúcnosti. Všetky konštantné časové intervaly spracovania, posielania packetov atď., spomenuté v tomto projekte boli stanovené a korigované práve procesom testovania.

5.2 Záťaž na výpočtový výkon



Obrázek 5.1: Záťaž na výpočtový výkon

Na obrázku číslo 5.1 je zobrazený graf, ktorý dokumentuje využitie procesora oboma programami. Konfigurácia systému bola nasledovná:

Tabulka 5.1: Konfigurácia systému

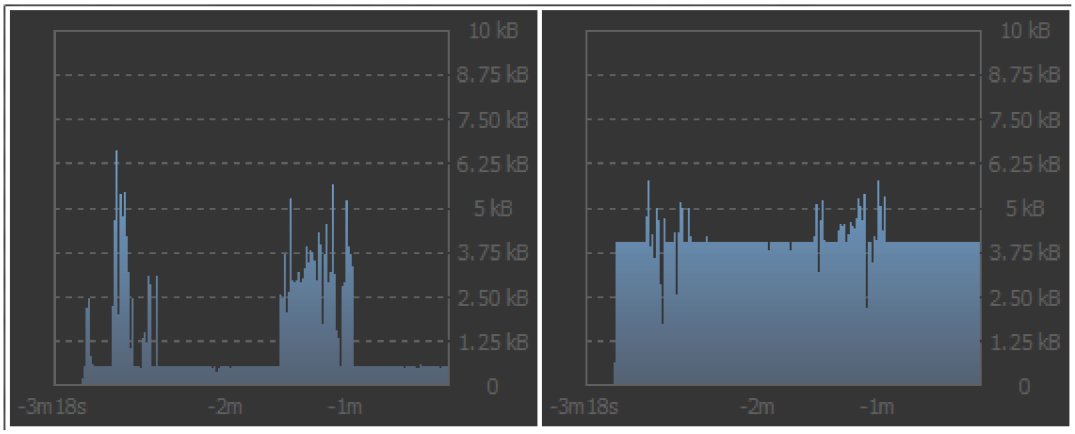
procesor	Intel Core 2 Duo E8400 3.2 Ghz
základná doska	Gigabyte Ultra Durable 3
operačná pamäť	Patriot 4GB Viper Extreme (kit 2x 2GB) 800MHz
grafická karta	ZOTAC GeForce GTX 260
operačný systém	Windows 7

5.3 Záťaž na sieťové zdroje

Obrázok číslo 5.2 dokumentuje využitie šírky pásma siete za behu aplikácie. Je jasne vidieť, že využitie siete je za behu programu naozaj minimálne.

5.4 Dotazník pre testerov

Pre zlepšenie a spresnenie vývoja v budúcnosti bol zostavený dotazník pre hráčov, ktorý hru pomáhali testovať.



Obrázek 5.2: Záťaž na sieťové zdroje. Na obrázku je zobrazené zaťaženie sieťe programom SMServer.exe s jedným aktívnym klientom. Ľavý graf predstavuje download, pravý upload.

Otázka č.1: Ako sa vám páčila hra?

Tester 1: „Myslím si, že hra Hon na ponorku, je jednoduchou sieťovou hrou, ktorá poskytuje hráčovi krátkodobé pobavenie, páčila sa mi najmä celkom presná detekcia kolízii pri strete objektov.“
Tester 2: „Hra vo mne zanechala pozitívny dojem. Keďže som sa na testovaní zúčastňoval od začiatku, oceňujem najme sieťový kód, grafická stránka by mohla byť lepšia.“
Tester 3: „Ak zoberiem do úvahy, že cieľom bolo vytvoriť jednoduchú hru, tak myslím, že výsledok je ok, i keď určite má svoje muchy.“

Otázka č.2: Aké má hra nedostatky, čo by ste zlepšili?

Tester 1: „Určite by to bola v prvom rade grafika. V prípade zlepšenia, by výrazne stúpol pozitívny dojem.“
Tester 2: „Hra sa mi páčila taká aká je i keď ako som spomenul, grafika by určite mohla byť lepšia.“
Tester 3: „Chcelo by to hlavne zložitejšie objekty podľa môjho názoru.“

Otázka č.3: Aké ma hra pozitíva?

Tester 1: „Ako som spomínal, zaujala ma hlavne presná detekcia kolízií objektov, sieťový kód sa zdá byť tiež celkom dobrý.“
Tester 2: „Páči sa mi ako programátorovi najmä riešenie serverovej časti a sieťového protokolu.“
Tester 3: „Veľmi sa mi páčila presná detekcia kolízie ponoriek a ostrovov. Čo sa týka hrateľnosti, tak určite bol dobrý nápad viac torpédových komôr.“

Otázka č.4 Aký výsledný dojem vo vás hra zanechala?

Tester 1: „Hra ako celok vo mne zanechala celkom dobrý dojem, ale ako som spomínal,

pekná grafika by ho výrazne dvihla.“
Tester 2: „Hranie hry vo mne zanechalo príjemný pocit, ale pravdou je, že kôli jednoduchosti za chvíľu zábava opadne.“
Tester 3: „Myslím si, že hra je zábavnosťou na úrovni starých arkádových hier, tak že po odohratí pár hier to vnímam podobne.“

Kapitola 6

Záver

Pri počiatočnom plánovaní, vytváraní návrhu a následnom rozvrhu pre jednotlivé časti sa rátalo s jednoduchou dvojrozmernou grafikou. Výber prostriedkov bol však viazaný na čo najväčšiu rozšíriteľnosť a preto sú zvolené knižnice DirectX tým pravým riešením. Smer, ktorým by sa mal budúci vývoj uberať určite zohľadňuje podstatné zlepšenie grafiky, poprípade prechod do tretieho rozmeru. Keďže však táto práca nie je orientovaná na grafický vývoj, je táto činnosť v budúcnosti až druhoradá.

Hlavným cieľom bolo vytvoriť sieťovú komunikáciu na kvalitnej úrovni, schopnú reálnej prevádzky, čo sa v konečnom dôsledku podľa plánu podarilo. Na základe uvedených zdrojov bol vytvorený návrh, ktorý bol neskôr implementovaný a následne testovaný. Finálny program zohľadňuje všetky kritériá, stanovené na začiatku a obsahuje mnoho rozpracovaných častí, ktoré môžu byť využité v rámci budúceho vývoja.

Po preštudovaní uvedených zdrojov a analýze techník, používaných v hrách, je vytvorený návrh, následná implementácia a testovanie výsledkom tejto práce. Všetky prevzaté nápady boli adekvátne upravené pre potreby tohto projektu. Výsledný zdrojový kód je vlastná tvorba, nie kompilát dostupných riešení.

Pre zlepšenie je výsledky tejto práce možné efektívne kombinovať s ostatnými prácami, zaoberajúcimi sa grafickou, hernou alebo sieťovou problematikou, vypracovanými v tomto roku. Napríklad, spracovanie projektu „3D svět“by bolo možné využiť pre zlepšenie grafickej stránky projektu. Projekt s názvom „Umělá inteligence pro deskovou hru“by mohol byť použitý pre tvorbu umelej inteligencie a počítačom ovládaných oponentov v tejto hre.

6.1 Plán pre grafický vývoj

Keďže celý grafický model je vytvorený pomocou nástroja na tvorbu trojrozmernej grafiky, určite by bolo vhodné adekvátne to využiť a dodať hre netriviálny trojrozmerný efekt. Ako už bolo spomínané v predchádzajúcich častiach, celá scéna je namodelovaná v 3D a zobrazovaný dvojrozmerný efekt vzniká zo statickej kamery a nepoužívanej súradnice z. Všetky programové štruktúry sú však uspôsobené na jej použitie. Určite by zlepšilo dojem z hry aj využitie textúr na jednotlivé modely objektov, čo by zas zdvihlo latku o niekoľko úrovní, poprípade využiť pre tvorbu nových zložitejších objektov profesionálny modelovací nástroj, ako napríklad 3D Studio Max alebo opensource Blender. Tieto úkony však exponenciálne zvyšujú náročnosť celého projektu a sú skôr výzvou pre celé tímy programátorov, nie jednotlivca. Budúci vývoj v skratke:

- Nové modely, vytvorené v špecializovanom nástroji

- Prechod do tretieho rozmeru
- Pridanie textúr, hmly, vĺn a iných grafických efektov
- Prechod na režim plnej obrazovky
- Pridať algoritmus interpolácie polohy objektov

6.2 Plán pre vývoj logiky aplikácie

Návrh a využitie vzťahov medzi programovými štruktúrami je silnou stránkou projektu. V budúcnosti umožňuje jednoduché pridávanie objektov bez výrazných znalostí celej architektúry. Pre prípad vývoja aplikácie je predpripravená trieda pre dynamické mapovanie kláves, ktoré by eventuálne mohol chcieť užívateľ zmeniť. Vhodné by bolo následné vytvorenie konfiguračného súboru pre uloženie nastavení a ich opätovné načítanie. Aktuálne je projekt vo verzii 1.0. Pri väčšom rozšírení by určite potreboval vyprecizovať a to hlavne:

- Pridanie máp
- Vytvorenie editora máp
- Nové zbrane, vylepšiť algoritmus navádzania torpéd
- Nové typy ponoriek
- Nové prekážky s dynamickou polohou
- Umelú inteligenciu a hru pre jedného hráča
- Umožniť užívateľovi „premapovať“klávesy

6.3 Plán pre vývoj sieťovej komunikácie

Základným cieľom na začiatku tohto projektu bolo vytvoriť dobrý sieťový protokol a následne kvalitnú klient-server komunikáciu. Po zhodnotení počiatočného plánu na konci prvej etapy vývoja si myslím, že tento cieľ sa podarilo splniť. Možnosť zlepšenia však vidím v posielaní správ zo servera smerom na klienta. Miesto aktuálnej pozície všetkých objektov a kolízií by bolo potrebné implementovať „delta compression“a posielať iba zmeny od posledného stavu. V aktuálnom stave to síce nie je vôbec problém, ale pri rádovo desiatkach až stovkách objektov by mohlo dochádzať k incidentom. Budúci vývoj siete v skratke

- „delta compression“

Literatura

[1] GameDev.net [online]. `http://www.gamedev.net/`, 2011.

[2] Valve Developer Community [online].
`http://developer.valvesoftware.com/wiki/Main_Page`, 2011.

[3] Barron, T.: *Multiplayer game programming*. Prima Tech's game development, Prima Tech, 2001, ISBN 9780761532989.
URL http://books.google.com/books?id=A9Z69ql3HrsC

[4] Bettner, P.; Terrano, M.: 1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond [online].
`http://www.gamasutra.com/view/feature/3094/1500_archers_on_a_288_network_.php`, 2001.

[5] Fielder, G.: Glenn Fiedler's Game Development Articles and Tutorials [online].
`http://gafferongames.com/`, 2011.

[6] Gibová, Z.: Pohyb telesa vo viskóznej tekutine [online].
`http://people.tuke.sk/zuzana.gibova/pohyboverovnice/pohybvisk.htm`, 2007.

[7] McShaffry, M.: *Game coding complete*. Safari Books Online, Charles River Media, 2009, ISBN 9781584506805.
URL http://books.google.com/books?id=ja6JPwAACAAJ

[8] Menard, M.: *Game Development with Unity*. Course Technology Ptr, 2011, ISBN 9781435456587.
URL http://books.google.com/books?id=WqK3cQAACAAJ

[9] Otto, G.: Rendering Approaches [online]. 2002.
URL http://viz.aset.psu.edu/gho/sem_notes/color_3d/html/rendering.html

[10] Parziale, L.; Britt, D.; Davis, C.; aj.: *TCP/IP Tutorial and Technical Overview*. Vervante, 2006, ISBN 9780738494685.
URL http://books.google.com/books?id=RnP4PAAACAAJ

[11] Postel, J.: RFC 768 [online]. 1980.
URL http://www.ietf.org/rfc/rfc0768.txt

[12] Postel, J.: RFC 793 [online]. 1981.
URL http://www.ietf.org/rfc/rfc793.txt

Příloha A

Obsah CD

+ **Hra** - spustiteľné súbory hry Hon na ponorku
+ **Návod** - krátky návod na preloženie projektu
+ **Zdrojové súbory** - zdrojové súbory hry Hon na ponorku
+ **Text** - obsahuje text práce v zdrojovom formáte (LaTeX) a tlačiteľnú (PDF) verziu

Příloha B

Slovník pojmov

AI umelá inteligencia

API skratka pre application programming interface

cd skratka pre cooldown

co-op herný mód, v ktorom hráč spolupracuje s ľudským partnerom a súťaží proti počítaču

cooldown čas, ktorý musí uplynúť, aby udalosť mohla znovu nastať

CTF herný mód, v ktorom hráč stráži vlajku svojho tímu a snaží sa získať vlajku oponentov

deathmatch herný mód, všetci hráči bojujú proti sebe

debugger vývojový nástroj na ladenie programov

„debug“ verzia verzia softvéru určená pre ladenie

domination herný mód, úlohou hráča je spolu s tímom kontrolovať čo najväčšie možné územie na mape

frame dátový packet na linkovej vrstve OSI modelu

GUI grafické užívateľské rozhranie

MMORPG typ hry, v ktorej je naraz v jednom hernom svet aktívnych tisíce hráčov

multiplayer hra videohra určená spravidla pre dvoch a viacerých hráčov

packet dáta, ktoré sú prenášané v celku

PvE skratka player versus enviroment, hráči bojujú spolu proti počítačom kontrolovaným oponentom

PvP skratka player versus player, hráči bojujú proti sebe

P2P označenie pre Peer-to-peer sieťový model

„release“ verzia verzia softvéru určená pre koncového užívateľa

runtime prostredie základný balík nástrojov, ktoré je nutné mať nainštalované pre beh aplikácie

singleplayer hra videohra pre jedného hráča, ktorý súťaží proti počítačom riadeným oponentom

spawn objavenie hráča na novej pozícii v prípade začiatku hry alebo zničenia

team deathmatch herný mód, hráči tvoria tímy, ktoré bojujú proti sebe