VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

HERNÍ SERVER PRO PODPORU ON-LINE HER

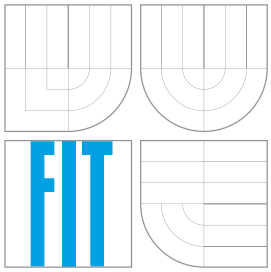
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN OHNHEISER

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

HERNÍ SERVER PRO PODPORU ON-LINE HER

GAME SERVER FOR ON-LINE GAMING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN OHNHEISER

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MARTIN HRUBÝ, Ph.D.

BRNO 2011

Abstrakt

Bakalářská práce pojednává o vývoji a tvorbě aplikace herního serveru pro podporu on-line her. Zabývá se klasifikací existujících podobných aplikací, síťovým rozhraním a herní logikou. Dále ukazuje možné použití knihovny ENet, SDL a OpenGL. Demonstruje funkčnost navrženého serveru na jednoduchých hrách.

Abstract

Bachelor's thesis deals with development and creation application of Game Server for Online Gaming. It deals with the classification of existing similar applications, network interface and game logic. It shows the possible use of library ENet, SDL and OpenGL. It demonstrates the functionality of the proposed server on simple games.

Klíčová slova

Herní server, on-line hry, hry více hráčů, MMO, Enet, OpenGl, SDL, klient-server, UDP.

Keywords

Game server, on-line games, multiplayer games, MMO, Enet, OpenGl, SDL, client-server, UDP.

Citace

Jan Ohnheiser: Herní server pro podporu on-line her, bakalářská práce, Brno, FIT VUT v Brně, 2011

Herní server pro podporu on-line her

Prohlášení

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

.....
Jan Ohnheiser
18. května 2011

Poděkování

Rád bych poděkoval svému vedoucímu Ing. Martinu Hrubému, Ph.D. za ochotu a čas strávený při konzultacích této práce. Také děkuji své rodině za stálou podporu při studiu.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Informační technologie zaznamenávají v současné době nevídaný rozvoj a uplatňují se snad ve všech oborech lidské činnosti. Služby informačních technologií mají své stále větší místo v rámci světové ekonomiky. Staly se nepostradatelným pomocníkem při práci, ale slouží také k zábavě a poučení. Nepopiratelnou součástí tohoto růstu jsou i počítačové hry.

Původ her spadá do daleké historie ještě před vznikem počítačové technologie, nejdříve to byly hry související se způsobem života, převažovaly hry bojové nebo zaměřené na lov a zajištění potravy. S technickým rozvojem se však měnil i způsob a podoba her. Příchodem počítačů slovo hra dostalo nový význam.

Rozvoj her zažívá dnes obrovské pokroky. Hry slouží nejen pro pobavení, ale i pro rozvoj představivosti či jiných dovedností.

Počítačová technika dodala hrám mnoho nových možností, hranice fantazie není zdaleka vyčerpaná. Pravidla her jsou stále složitější a počítače stále výkonnější. Hry jsou postaveny na základě interakce hráče s počítačem, a v případě síťových her se jedná o spolupráci mezi mnoha počítači, umožňující tak hru více hráčům. Komunikace po síti probíhá na velké vzdálenosti i mezi světadíly, tímto se počet spoluhráčů, hrajících v jednom okamžiku jednu hru, mnohonásobně zvyšuje.

Cílem bakalářské práce bylo prozkoumat již existující on-line hry a systémy pro interakce her s uživateli. Jsou zde popsány různé druhy her a jejich možná dělení podle žánru či řízení času a dalších kritérií. Práce pojednává i o různých architekturách stavby a spojení komunikace mezi uživateli. Dále se pak zabývá návrhem a tvorbou samotného herního serveru pro podporu on-line her, který umožňuje napojení herních klientů, dále i návrhem a tvorbou těchto klientů. Nakonec i předvedení funkčnosti herního serveru na vytvořených demonstračních hrách, využívajících jeho funkcí, a následné testování těchto her v reálném provozu.

Kapitola 2

O hrách více hráčů

Hry, ať už stolní, karetní nebo třeba i sporty, jsou často zábavnější právě za účasti více hráčů. Proto rychle vznikly první počítačové hry komunikující po síťovém rozhraní. Další vývoj rozvětvil rozsáhlý strom všemožných her. Nové čerpají ze starých, staré čerpaly ze starších.

2.1 Rozdělení her

Existuje mnoho her, všechny se navzájem různě liší. Jejich jednoznačné rozdělení do skupin neexistuje, proto jsou řazeny do skupin podle různých aspektů. Zde jsou uvedeny některé z nich.

2.1.1 Rozdělení podle řízení času

Reálný čas ve hrách nemusí vůbec nic znamenat. Tím míň ve hrách, které pracují na číslicových počítačích. Z pohledu toku času lze hry rozdělit na:

- Turn-based (tahové hry) – Reálný čas není důležitý, důležité jsou pouze tzv. tahy. Tahy nahrazují časovou jednotku, obvykle může hráč provést omezený počet operací během jednoho tahu. Příkladem tahových her mohou být například piškvorky či šachy.
- Real-time (hry v reálném čase) – Herní čas v těchto hrách napodobuje reálný čas, obvykle jeho přímou úměrou. Hráč musí zvládnout co nejrychleji reagovat na nastávající situace. Příkladem mohou být závodní hry či bojové simulátory a další.

Některé hry kombinují více různých toků času (například tok časů kombinuje série válečných strategií Total War [4] – čas na taktické mapě jde v tazích, kde se pohybují celé armády, samotná bitva však běží v real-time módu). To však nelze dobře aplikovat ve hrách s více hráči. Ostatní, kteří se neúčastní události, jenž změnu toku času vyvolala, musí počkat než zase skončí, aby byl zachován časový kontext. Často je tento problém řešen vypuštěním částí hry tak, aby zbyl právě jeden tok času pokud možno s tou zajímavější částí hry (hráči v Total War mohou měřit své síly na bitevním poli v real-time).

2.1.2 Rozdělení podle žánru

Žánr hry je velmi důležitým aspektem, naskytne nám rychlý náhled, co lze od hry čekat. Mnoho lidí si oblíbí právě jeden či více žánrů a podle toho řídí svůj výběr her.

- Adventury – Jsou zaměřené více na příběh a logiku. Hráč řídí kroky postav a přitom řeší spletité hádanky a zápletky. Žánr se objevil už v 80. letech 20. století, nejdříve v textových podobách. Hrající si přečetl situaci a podle toho zadával textové příkazy ve smyslu prozkoumat místnost, sebrat předmět, zatlačit na páku apod., což vyvolávalo další situace. Postupně tento žánr přešel i do grafické stránky. Jako právě takovým příkladem může být česká hra *Machinarium* (viz [1]). Online hry obvykle nebývají přímo adventury, ale často se vyskytují prvky adventur v kombinaci s jinými žánry.

- Akční hry – Typicky sem patří střílečky (First person shooter). Cílem bývá probojovat se řadami nepřátel na rozličných mapách do určeného místa. Spadají sem hry jako je *Half-Life* a jeho neodmyslitelná modifikace *Counter-Strike*.

- Arkády – Sem patří sportovní, závodní, logické a skákací hry. Často jednodušší nápad, provedení, ovládání i příběh, má však něco do sebe s řešením levelů se stupňovanou obtížností. Příkladem může být i hra *Atomix*, *Pac-Man* či *Arkanoid*.

- Strategie – Má zacíleno na vůdčí schopnosti hráče, jehož úkolem je řídit nějaký větší celek (zemi, město, tým atd.) k pokud možno co nejlepšímu výsledku. Původně převažovaly strategie tahové či budovatelské. Příchodem *RTS* (Real-Time Strategy) hry *Dune 2* (článek o hře [7]) se však pohled na strategie změnil k dnešní podobě.

- Simulátory – Jak již název napovídá, hra simuluje reálné objekty a prostředí. Hráč řeší úkoly, které by mohly nastat v normálním životě, ale obyčejně se k nim nedostane. Simulátory mohou být prakticky na cokoli, co člověka napadne. Od simulátoru venčení psa až k vesmírným bitvám.

- Hry na hrdiny (Role-Playing Game) – Obvykle zasazené do středověkých či fantasy příběhů, ve kterých je cílem plnění množství různorodých úkolů, mnoho z nich ani nemusí souviset přímo s dějem. Hráč ovládá hlavní postavu nebo více hrdinů, kteří v průběhu hry získávají postupným plněním úkolů nové dovednosti. Tento žánr se stal největším kandidátem pro tvorbu *MMOG*.

Toto rozdělení není nejpřesnější, jelikož jsou žánry ve hrách mezi sebou různě promíchávány a vznikají tak zajímavé kombinace.

2.1.3 Rozdělení na dedikované a listen servery

Dalším pojmem jsou tzv. *Dedikované servery* a *Listen servery*, rozdíl zásadně ovlivňuje celý propojovaný systém.

Dedicated servers (vyhrazené servery)

Server a klient jsou dvě odlišné aplikace navzájem komunikující. Úkoly mají jednoznačně rozděleny. Zatímco server hostuje a řídí hru, klienti interagují s hráči u svých počítačů. Server se tak nestará o žádného hráče a nemusí se zabývat vykreslováním, jenž by jej zbytečně zatěžovalo. Ve výsledku pak může pojmout větší počty hráčů (*MMOG* bývají právě tohoto typu, popsáno níže na 2.1.5).

Listen servers (naslouchající servery)

V tomto případě server hostující hru pracuje zároveň jako plnohodnotný klient. Jedna aplikace v sobě obsahuje jak klienta, tak i server. Po vytvoření hry je hráč, který hru vytvořil, automaticky připojen k nové hře, a funguje jako další klient. Jakožto další hráč však nemá žádné výhody, kromě toho, že pro něj neexistuje zpoždění paketů na síti plus nějaké administrátorské příkazy. Listen server má nevýhodu v tom, že nelze použít pro větší počty připojených klientů, protože musí ještě obstarávat i obsluhu svého hráče se vším všudy. Listen server je však uživatelsky přívětivější. V kombinaci s centrální databází založených her, kam se servery registrují, pak umožňuje připojení i bez znalosti IP adresy serveru.

2.1.4 Webový klient a Aplikační klient

Webový klient může být výhodou i nevýhodou. Oproti aplikačnímu klientovi není instalován na PC, ale musí být po každém přihlášení stažen ze serveru, je však dostupný ze kteréhokoliv místa připojeného na internet. Z hlediska programovacích jazyků webový klient může používat HTML, XHTML, Java, Javascript, PHP, Flash, MySQL atd.

Aplikační klient naopak musí být předem instalován na PC. To však dává velkou výhodu v tom, že aplikační klient má většinu dat už zabudovanou v sobě, nepotřebuje žádné přebytečné data, jako jsou modely, animace, textury, dokonce může mít u sebe nainstalován dopředu i celý herní svět. Ten pak může být mnohem detailnější a vůbec tím nezvyšuje zátěž serveru. Posílaná data jsou pak jen ty, které se dynamicky mění (pohyby postav, objektů, jejich jiné akce atd.).

2.1.5 Další možná kritéria dělení

V této části jsou popsány další pojmy, jenž mohou přiblížit předběžný náhled na hru.

MMOG

Dělení podle počtu možných hrajících hráčů. Pokud uvažujeme hry s více hráči po internetu, naskýtá se nám pojem *MMOG* (massively multiplayer on-line game – masivní vícehráčská on-line hra). *MMOG* je označení her, ve kterých mohou spolu hrát stovky i tisíce hráčů. Typicky sem patří *MMORPG* (massively multiplayer on-line role-playing game).

Příkladem *MMORPG* jdou firma *Blizzard s hrou World of Warcraft* (na [5]), jenž se svým enormním počtem aktivních hráčů přesahuje hranici 12 milionů, či další hry jako *RuneScape* (na [3]) a mnoho dalších.

Kooperativnost

Dalším aspektem může být dělení na *kooperativní hry*, kde hráči spolupracují na nějakém větším společném úkolu, nebo na *konkurenční hry*, jejichž hráči mezi sebou soutěží o získání nejlepšího výsledku právě pro sebe. Obvykle také nelze rozdělit hry přímo na tyto dvě skupiny, prolínají se v podobě týmů či aliancí (skupinám hráčů) bojujících proti sobě.

2.1.6 Shrnutí

Existuje mnoho různých způsobů rozdělení, výše uvedené jsou jen některé z nich. Tato práce je zaměřena právě na vývoj dedikovaného serveru s architekturou server-klient a aplikačních

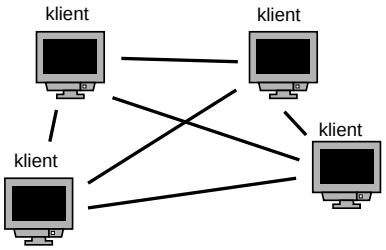
klientů. Žánr bude nezávislý na knihovnách vyvíjeného serveru, určí jej teprve specifikace výsledné aplikace. Také bude záležet na cílové aplikaci, jak bude potřeba řídit chod herního času. Server bude podporovat jak tahové, tak real-time hry.

2.2 Architektura on-line her

Jakékoliv aplikace, které komunikují mezi sebou po síti, musí mít nějaký řád, architekturu síťových spojů. Můžeme se setkat se dvěmi základními druhy propojení většího množství aplikací, jsou to *peer-to-peer*, kde mají všichni stejné pravomoce, nebo *klient-server*, kde je celý systém řízen jedním centrálním prvkem – serverem [8]. Při návrhu síťových aplikací můžeme zvolit obě možnosti.

2.2.1 Peer-to-peer

Architektura založená na vzájemné důvěře všech uživatelů. Všechny uzly jsou si rovny. Počítače mezi sebou komunikují jak zrovna potřebují (viz obr. 2.1), každý z nich pracuje s vlastními daty. Při použití v jednotném systému hry však musíme určit, kdo která data bude vlastnit a spravovat, následně je pak sdílet ostatním klientům. V Peer-to-Peer musí všichni klienti držet informace o spojení s ostatními. Pokud by se odpojil klient, jenž by byl spojovacím uzlem s ostatními, rozdělila by se hra na dvě nezávislé hry.



Obrázek 2.1: Architektura Peer-to-peer

Výhody Peer-to-peer:

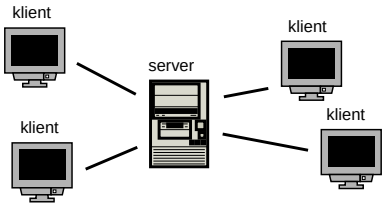
- Rozdělení zátěže mezi klienty.
- Po pádu libovolného prvku není nic závažného ztraceno, hra může pokračovat dále.

Nevýhody Peer-to-peer:

- Počet komunikačních kanálů (každý s každým) mezi klienty roste s druhou mocninou počtu klientů.
- Nutnost zamezit kolizí dat, kdy se aplikace neshodnou.
- Data na klientech lze podvrhnout, následně pak rozesílat chybová data ve svůj prospěch ve hře.
- Bez možnosti centrální autentizace hráčů.

2.2.2 Klient-server

Jak již název napovídá, tato architektura se skládá ze serveru a klientů postavených obvykle do hvězdicové soustavy se serverem jako středem (viz obr. 2.2). Jsou to tedy dvě odlišně pracující aplikace. Server obstarává veškerý chod hry, registruje příchozí a odchozí klienty a plní jejich příkazy. Klient interaguje s hráčem a jeho akce přeposílá serveru. Od něj pak dostane nazpátek informace o stavu hry, jenž zase klient ukáže hráči na monitoru.



Obrázek 2.2: Architektura Klient-server

Výhody Klient-server:

- Každý klient komunikuje většinou pouze se serverem.
- Data jsou centralizována na serveru, ten jednoduše řídí procesy s daty.
- Server má vždy pravdu, pokus o podvrhnutí dat může být jednoduše a efektivně zlikvidován.
- Centrálně řízená autentizace hráčů.

Nevýhody Klient-server:

- Zátěž se soustřeďuje na server.
- Pád serveru odpojí i všechny klienty ze hry.

2.2.3 Shrnutí

Pro on-line hry se častěji používá právě architektura klient-server. Pro takovéto účely je rozhodně výhodnější díky možnosti centralizovaného řízení. Vyvíjený projekt se také neodchýlil od tohoto konceptu. Cílem je právě architektura klient-server.

2.3 Možnosti ukládání dat hráčů

Některé hry přímo vyžadují uložení dat hráčů, jejich postav a předmětů. Jedná-li se o hru, ze které je možno se v průběhu odhlásit, a po následném přihlášení má být vše ve stejném stavu, v jakém byl před odhlášení, je nutné mít možnost i úschovy takových dat. Můžeme zvolit mezi použitím databází nebo přímý zápis a čtení ze souborů.

Přímé čtení a zápis v souboru je rychlá záležitost, není nutná žádná speciální knihovna a implementace je jednoduchá. Musíme ale dobře navrhnout strukturu souborů a práci s nimi. V případě větších počtů hráčů vzniká problém vyhledání správného místa v souboru.

Další možností je připojení databáze, často to bývá SQL nebo MySQL databáze. Použití je jednoduché, lze snadno přidávat a odebírat prvky, měnit data či pořizovat statistiky. Databáze jsou však ve výsledku pomalejší než přímý přístup k souboru a potřebuje k chodu speciální knihovny pro podporu databází.

Tyto možnosti nejsou přímo v knihovnách herního serveru rozváděny, nelze totiž předem odhadnout, jaké data, v jakém rozsahu a jestli vůbec bude potřeba ukládat.

Při užití ukládání dat hráčů je také nutné zajistit bezpečnost těchto dat. Implementace zabezpečení autentizace pomocí hesel je k tomu nezbytné.

2.4 Herní prostředí

Od dob vzniku prvních grafických karet se začaly objevovat hry s grafickým prostředím. Textové hry postupně upadaly a byly nahrazeny grafickou konkurencí. Nové možnosti umožnily tvorbu a zobrazení zajímavých herních světů, po nichž se hráči pohybují. Ať už dvourozměrné či třírozměrné světy se staly jedním z důležitých faktorů hodnocení her.

Hry lze rozdělit na ty, jenž se snaží herním světem přiblížit se co nejblíže reálnému, a na ty, jenž nechávají více prostor fantazii.

Při pohledu na herní prostředí z hlediska jeho tvorby a potřeb, jenž jsou požadovány, lze rozpoznat několik používaných způsobů tvorby světů.

2.4.1 Rastrové prostředí

Svět je složen z dvourozměrné, někdy i třírozměrné matice. Velikost buněk není nijak omezena, jen v rámci hratelnosti. Každá buňka takové matice nabývá hodnot jako kámen, voda nebo volné pole, apod., tím pak dohromady tvoří rozmanité prostředí. Jednoduše lze vytvářet pravoúhlé útvary, avšak šikmé nebo oblé tvary mohou být problém.

Jedno čtvercové pole má osm směrů k okolním polím, ale šikmé směry mají jinou vzdálenost. V důsledku vyvážení směrů se ve hrách vyskytují i šestiúhelníková pole. To ale zase zamezuje používání pravých úhlů.

Použití rastrového prostředí je velmi užitečné, pokud je nutné provádět dynamické změny v terénu, jedná se o pouhé nastavení potřebných buněk na potřebnou hodnotu a změna bude hotova.

Obvykle se ve hrách vyskytují dvourozměrné matice. Dobrým příkladem použití tohoto způsobu ve tří rozměrech je hra *Minecraft* (bližší informace v [2]), v níž je zakomponován i velmi dobře zpracovaný generátor map.

2.4.2 Volné vektorové prostředí

Tentokrát zde není žádná matice, terén se skládá z jednotlivých polygonů, z nichž každý jejich bod je dán svou vlastní pozicí. Nezáleží na tom, zda je položen ve dvou či třech rozměrech. Do prostředí lze zakomponovat jakýkoli útvar. Zvládne pracovat i s oblými tvary. Dynamické změny terénu jsou však mnohem složitější.

Většina dnešních her používá právě takovéto prostředí právě díky jeho možnostem tvorby libovolných tvarů. Obvykle lze prostředí modelovat v profesionálních modelovacích programech a následně jej exportovat do hry.

2.4.3 Height-map prostředí

Další možnost je *height-map*. Jedná se o jakousi kombinaci mezi rastrem a vektorem. Ve třírozměrném prostředí je definována dvourozměrná matice, jenž má v každé buňce uloženo výškové položení daného bodu. Terén lze s použitím *height-map* velmi efektivně generovat různými funkcemi, vznikají tak i terény blížící se realitě.

Height-mapu lze velmi jednoduše dynamicky upravovat. Pozice bodů je ale možné posouvat jen v jedné ose souřadnic. Ideální na vytvoření kráterů. Zato ale nelze dostat dva body z jedné height-mapy nad sebe. nepoužitelné například při pokusu v terénu vykopat tunel.

Na height-mapu se ve většině případech přidávají pomocné statické objekty, jako jsou stromy, skály či budovy. Samotná mapa je bez nich plochá a nudná, to pak není pro hratelnost dobré.

Také je často používána pro své vlastnosti při generování a následné konvertování do volných prostředí, po převedení na vektorový model však ztrácí své deformační možnosti.

2.4.4 Shrnutí

Všechny tři druhy prostředí jsou ve hrách hojně používány, každé se hodí pro něco jiného. Některé hry vyžadují dynamičnost terénu, proto vybírají maticový tvar v němž lze hodnoty buněk lehce měnit, jiné hry chtějí volné modelované a statické prostředí.

Kapitola 3

Návrh a implementace herního serveru

Tato kapitola obsahuje teoretické informace potřebné při návrhu vlastních knihoven pro hry s více hráči, jejich samotný návrh a implementaci.

3.1 Náhled na cíl

Cílem návrhu serveru je knihovna pro servery s podporu on-line her. Jedná se o dedikovaný typ serverů s architekturou klient-server. Nebude zaměřen na žádný předem určený žánr, v tomto ohledu zůstane otevřený pro výslednou aplikaci. Bude schopen obsluhovat jak tahové, tak i real-time hry.

3.2 Nástroje pro vývoj

Jako implementační jazyk byl použit C++ jazyk pro jeho efektivitu a rozsáhlé možnosti (bližší informace jsou čerpány z [6]).

3.3 Protokoly TCP/IP sítě

Způsob, jak donutit počítač komunikovat s ostatními. K tomu slouží rodina protokolů *TCP/IP*, jejichž model sestává ze čtyř vrstev (fyzická, síťová, transportní a aplikační, viz tabulka 3.1), jejichž hlavičky jsou hierarchicky uspořádány a postupně zaobalují potřebná data k doručení.

Fyzická vrstva využívá ethernetový protokol pro komunikaci mezi jednotlivými stanicemi a routery. Jeho hlavička skrývá MAC adresu potřebnou pro identifikaci jednotlivých zařízení po cestě k cíli. Na *síťové vrstvě* pak použijeme IP (internet protokol), jenž podle adresy v hlavičce jednoznačně nasměruje vyslaný packet na cílový počítač.

3.3.1 Protokol transportní vrstvy

Zásadní vrstvou TCP/IP je *transportní vrstva*. Správně zvolit protokol na této vrstvě je důležité pro celou síťovou komunikaci. Existují dvě základní možnosti výběru, a to TCP a UDP.

vrstva TCP/IP	protokol
Aplikační	vlastní
Transportní	UDP/TCP
Síťová	IP
Fyzická	ethernet

Tabulka 3.1: Model TCP/IP a příslušné protokoly

UDP (User Datagram Protocol)

Jedná se o jednodušší protokol *transportní vrstvy* TCP/IP (blíže v [9]). Jeho hlavička připojená k tělu obsahuje zdrojový a cílový port aplikace (použit pro adresování jednotlivých aplikací na počítačích), délku těla zprávy a kontrolní součet. Dále následují už holá data poslaná aplikací. Není zaručeno doručení odeslané zprávy ani pořadí příchodu zpráv. Komunikace použitím UDP je však rychlá a má nízké nároky na provoz po síti. Používá se hojně tam, kde není nutné doručit všechny zprávy, například ve video-konverzacích, kde ztracený paket neudělá žádné větší škody.

TCP (Transmission Control Protocol)

Je dalším kandidátem na použití pro *transportní vrstvu* TCP/IP. Oproti UDP je ale složitější. TCP hlavička (Podrobněji popsáno v [10]) obsahuje oproti výše zmíněnému protokolu navíc další řídící data (například číslo paketu, příznakové bity s různým nastavením, potvrzení paketu, ...). Komunikace vyžaduje ustavení spojení tzv. "handshake". TCP zajišťuje příchod paketů a doručení ve správném pořadí tak, jak byly odesílány. Na provoz jsou však větší požadavky. Proto se hodí pro aplikace, jenž požadují záruku doručení a ještě ve správném pořadí, například přenos textových dokumentů či binárních souborů.

Shrnutí protokolu transportní vrstvy

Pro naše účely potřebujeme co nejrychlejší interakci mezi serverem a klienty, nechceme však data ztrácet při přenosu či aby přicházela ve špatném pořadí. Řešení naskýtá zajímavá knihovna ENet, popsaná v následující kapitole.

3.3.2 Knihovna ENet

Jedná se o knihovnu v jazyce C (viz [11]) původně psanou pro akční střílečku z pohledu první osoby *Cube*. Později ale byla uvolněna pod otevřenou licencí pro volné užívání a modulaci.

Důvod vzniku této knihovny dala potřeba rychlé a spolehlivé komunikace po síti, kde data byla posílána v krátkých intervalech za sebou s požadavkem na nízkou latenci. UDP není dostatečně spolehlivé, neobsahuje žádné funkce řídící tok dat a TCP nezvládá zprávy v krátkých intervalech doručovat včas. Řešením může být právě ENet.

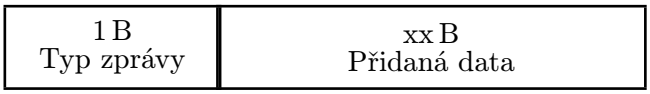
ENet slouží jako multiplatformní nadstavba UDP protokolu na transportní vrstvě. Poskytuje jednoduché rozhraní pro navázání komunikace po síti, sleduje životnost a stav spojení mezi stanicemi. Zajišťuje číslování paketů s doručováním ve správném pořadí. Umožňuje také vytvářet vícekanálová spojení, s různými prioritami spojení a nastavit nutnost doručení paketů. Pokud do přípustné doby nepřijde odpověď na odeslaný paket, odesílá se znovu, podobně jako u TCP. Zvládá posílat a přijímat zprávy s neomezenou velikostí, větší zprávy rozdělí na menší části a pošle ve více paketech.

Navíc se jedná o multiplatformní knihovnu, pracuje pod OS Windows i na Unix-like systémech.

V navrhovaném serveru byla tato knihovna použita pro navázání spojení a následnou komunikaci s klienty, musí s ní tedy pracovat i připojovaní klienti.

3.3.3 Vlastní tvar zprávy, aneb aplikační vrstva

Ještě zbývá protokol aplikační vrstvy TCP/IP. Zde už není použit žádný předepsaný protokol, ale je nutné přejít k vlastnímu, který obsahuje samotná data potřebná pro chod aplikace. Herní server si bude s klienty posílat velké množství různých zpráv, proto je nutné dobře navrhnout i protokol aplikační vrstvy. Samotná zpráva musí nést informace, tedy data a jejich význam. Jak návrh takovéto struktury může vypadat, je na obrázku 3.1.

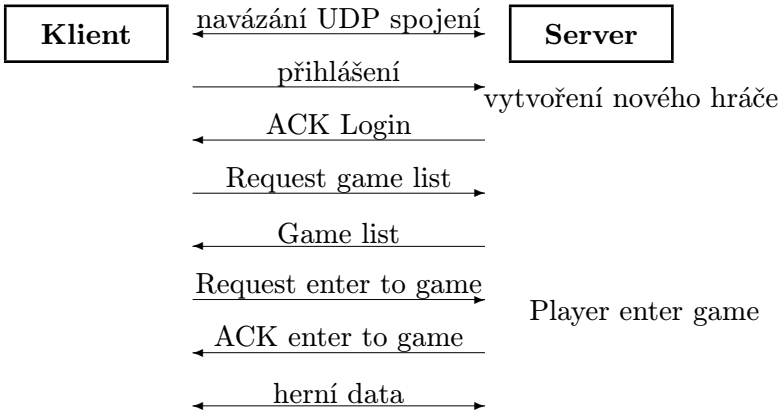


Obrázek 3.1: Jednoduchý aplikační protokol

Pokud se bude jednat o jednoduchou zprávu, data za typem zprávy nemusí být žádná, jinak mohou dosahovat libovolných velikostí. Pokud by nestačil 1 byte pro dostatečné určení typu zprávy, lze přidat jednoduchou konstrukcí dotaz na další byte.

3.4 Funkce herního serveru

Cílem je navrhnout herní server, jenž bude hostovat hry založené klienty. Bude fungovat jako dedikovaný server – sám se her nebude účastnit. Musí si ale uchovávat data spuštěných her, připojených hráčů a aktivně komunikovat s klienty. Komunikace bude vypadat přibližně následovně (viz obrázek 3.2).



Obrázek 3.2: Příklad komunikace klient-server

Klient naváže spojení na server, k tomu musí znát port aplikace a IP adresu počítače, na němž server běží. Potom může vyslat žádost o přihlášení, tj. zprávu obsahující jméno (popřípadně šifrované heslo, to ale není v serveru aplikováno). Po úspěšném přihlášení

dostává zpět potvrzení a s tím i možnost vstupu do her. Server zatím jeho informace se jménem uloží pro příští použití. Dále nesmí chybět možnost získat seznam existujících her, aby mohl zjistit, ke které je možné se připojit. Další možnou zprávou je požadavek na vstup do hry. Server na tento požadavek reaguje přidělením struktury s daty hráče k existující hře, do níž hráč požádal o vstup. Jednotlivé založené hry jsou pro jednoduchost indexovány čísly, ale každá hra má i název určený žádostí o novou hru. Hráč, který hru vytvořil, dostává k dispozici další příkazy, jako je ukončení hry, spuštění, restart či zastavení. Nemůže chybět i opuštění hry a odhlášení. Samotná herní data budou posílána po stejné cestě. Pro odlišení herních a řídících dat jsou použity dva úvodní bity z bytu určující typ zprávu, jsou-li nastaveny na 11 (tj. hodnota $\texttt{C0}_{16}$ a větší), zprávu zpracuje herní server, pokud ne, pošle ji do samotné hry, ke které je hráč připojen. Typy možných zaslaných zpráv jsou uvedeny v tabulkách 3.2 a 3.3.

Příchozí zpráva serveru			
Typ	**význam**	**data**	**požadavky**
0xc0	přihlášení	jméno	–
0xc1	odhlášení	–	přihlášení
0xc2	požadavek na stav přihlášení	–	přihlášení
0xc4	nová hra	název hry	přihlášení
0xc5	konec hry	–	vlastník hry
0xc6	vstup do hry	index hry	přihlášení
0xc7	odchod ze hry	–	účastník hry
0xc8	start/restart hry	–	vlastník hry
0xc9	zastavení hry	–	vlastník hry
0xca	žádost seznamu her hry	–	přihlášení
0xcb	žádost seznamu hráčů ve hře	–	účastník hry
0xcc	textová zpráva hráčům	text	účastník hry
0x00-0xbf	herní data	data...	účastník hry

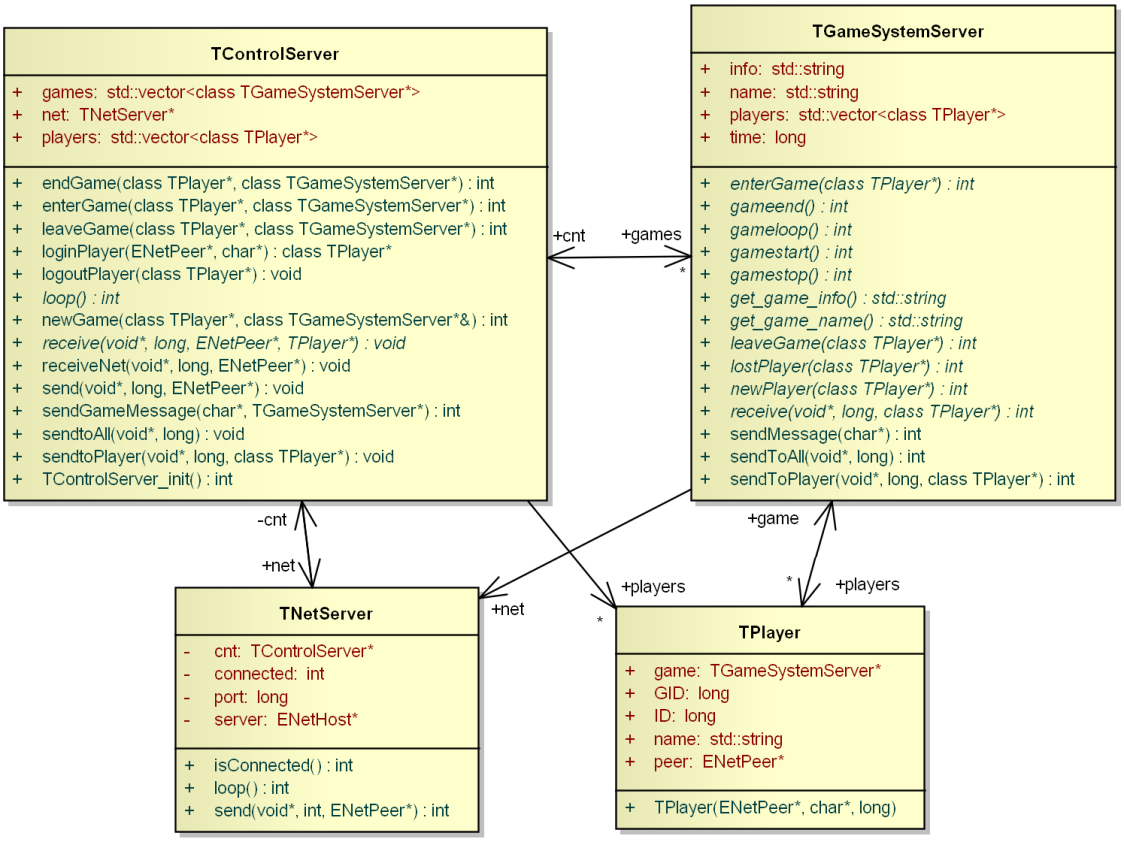
Tabulka 3.2: Seznam možných příchozích zpráv serveru

Odpověď serveru		
Typ	**význam**	**data**
0xc0	stav přihlášení	jméno, id, aktivní hra
0xc1	chyba: klient nepřihlášen	–
0xc2	chyba: klient není v žádné hře	–
0xc3	chyba: klient není vlastníkem hry	–
0xc4	ostatní chyby	–
0xc6	vstup do hry	index hry
0xca	seznam her	informace o hrách
0xcb	seznam hráčů ve hře	informace o hráčích
0xcc	textová zpráva hráčům	text
0x00-0xbf	herní data	data...

Tabulka 3.3: Seznam možných odpovědí serveru

3.5 Objektový návrh serveru

Na obrázku 3.3 je zjednodušený nákres objektové struktury herního serveru bez herních objektů. Tyto objekty a vazby mezi nimi budou vytvořeny až v konečné aplikaci herního systému dle potřeby dané hry.



Obrázek 3.3: Zjednodušený diagram tříd serveru

3.5.1 Jádro serveru

Jádro celého serveru tvoří třída `TControlServer`. Konkrétní aplikace využívající funkcí serveru může použít potomka této třídy jako rozhraní pro komunikaci mezi ostatními objekty a připojenými klienty. Zařizuje funkce pro posílání zpráv klientům, ať už chybových či potvrzovacích. Dále provádí kontrolu příchozích zpráv, pokud je první byte zprávy určující její typ větší nebo roven $C0_{16}$, zpracuje jí sám (Typy příchozích zpráv serveru jsou blíže popsány v tabulce 3.2), jinak ji přeposílá třídě `TGameSystemServer`, ve které se účastní klient posílající danou zprávu.

Přihlášení hráči se vkládají do vektoru tříd `TPlayer` spolu s ostatními daty. Po přihlášení je vždy jedna položka přidána a při odhlášení odebrána. Po příchodu požadavku o novou hru je vytvořena třída `TGameSystemServer` následně vložená do vektoru obsahující spuštěné hry. Pokud se ve hře nebude nacházet žádný hráč nebo bude přijat požadavek na ukončení hry od zakládajícího hráče, je následně hra odstraněna z vektoru a všichni hráči jsou z ní odpojeni.

Tato třída se dále stará i o periodickou práci jednotlivých her. V daných časových úsecích volá funkce herní smyčky všech her, v nich pak mohou pohybovat objekty a další nutné akce závislé na čase. Tahové hry tuto funkci nemusí používat.

Obsahuje totiž funkce pro komunikaci s klienty a reakce na jejich zprávy, například založení her, přihlašování hráčů a výpisy her (viz příkazy v tabulce 3.2). Samotnou komunikaci zpracuje třída `TNetServer`. Vlastní obsah paketu se však zjišťuje až v `TControlServer`, kde jsou zkontrolovány, případně jsou provedeny nebo poslány dál do samotné hry.

3.5.2 Síťové rozhraní

Jednou z tříd spravovaných třídou `TControlServer` je `TNetServer`, sloužící k práci se sítí. Tato třída používá funkce samotné knihovny ENet. Po vytvoření inicializuje ENet jako server přijímací spojení. Dále pak obstarává odesílání a přijímání zpráv a další případné události vyvolané ENet, jako je ztráta spojení. V takovém případě volá v `TControlServer` funkci odhlášení klienta, tím i jeho případné odpojení ze hry.

3.5.3 Data hráče

Je potřeba uložit data jednotlivých hráčů. K tomu slouží třída `TPlayer`, v níž je obsaženo jeho jméno, pod kterým se připojil, index hráče pro jeho identifikaci ve vektoru a také odkaz na hru, ke které je připojen. Každý hráč může být připojen maximálně na jednu hru, každý klient může být připojen maximálně jako jeden hráč. Tato třída nemá kromě konstruktoru a destruktoru žádné další funkce. Třída je vytvořena třídou `TControlServer`, právě když se klient přihlašuje, a je smazána při odhlášení či neočekávaném výpadku (pro zamezení vzniku zombií hráčů, jenž už nemají fyzické spojení se serverem).

3.5.4 Herní systém

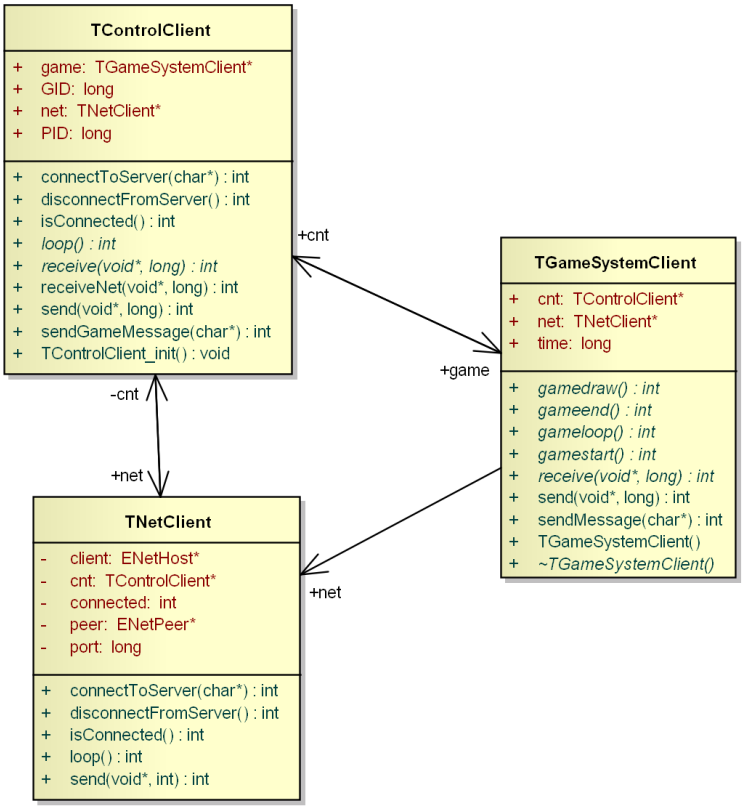
Další důležitou třídou je `TGameSystemServer`, jedná se o třídu reprezentující založené hry na serveru. Při každé žádosti o novou hru je jedna instance potomka této třídy vytvořena. Při tom se také nastaví i jméno zakládané hry. V případě ukončování hry je instance smazána. Tato třída má pro své potomky přichystané virtuální funkce pro přijímání, posílání zpráv a řízení hry, jako jsou příchody hráčů, pokyn startu či restartu, nebo odchody hráčů. Dále umožňuje posílání zpráv s daty hry klientům a naopak příjímání a třídění zpráv od klientů.

3.6 Objektový návrh klienta

K serveru je potřeba navrhnout i klienty, jenž budou s tímto serverem komunikovat. Na obrázku 3.4 je nákres zjednodušeného Diagramu tříd klienta.

3.6.1 Jádro klienta

Obdobně jako u serveru slouží i třída `TControlClient` k řízení celého systému. Stará se o třídy `TGameSystemClient` a `TNetClient`, obě budou mít právě jednu vytvořenou instanci (klient nemůže být současně připojen ve více hrách). Tato třída reaguje na přijaté zprávy a odesílá požadavky na server. Obsahuje funkce pro posílání jednotlivých požadavků, jako jsou přihlášení či zakládání her.



Obrázek 3.4: Zjednodušený diagram tříd klienta

3.6.2 Síťové rozhraní

Třída `TNetClient` je velice podobná třídě `TNetServer`, ale nevytváří pomocí ENet server přijímací spojení, nýbrž klienta navazujícího spojení se serverem.

3.6.3 Herní systém

Instance třídy `TGameSystemClient` se v klientu vytvoří jen jednou, je podobná jako herní systém v serveru, neobsahuje ale řídící funkce pro hru, pouze jen posílání a přijímání zpráv serveru a herní smyčku. Většinou virtuální metody, aby aplikace mohla definovat vlastní zákonitosti běhu aplikace.

3.6.4 Vizualizace a interakce s uživatelem

Aby klient mohl použít grafické zobrazení, bylo využito knihovny OpenGL. OpenGL (tutoriály a návody jsou k nalezení na [13]) je grafickou knihovnou použitou v klientech, podporuje práci ve většině používaných platformách.

Další použitou knihovnou je SDL (Simple DirectMedia Layer, blíže v [12]). Je to volně dostupná multiplatformní multimediální knihovna opatřující nízkoúrovňový přístup ke klávesnici, myši, joysticku, audiu, videu atd. SDL, napsaná v jazyce C, je kompatibilní s C++. Umožňuje i práci s grafickou knihovnou OpenGL. Právě proto byla zahrnuta do projektu.

Ke komunikaci klienta s hráčem byla za pomoci těchto dvou knihoven vytvořena třída

`TWindow`. Využívá knihovny SDL pro vytvoření grafického okna s podporou OpenGL, v klientu pak lze tedy použít i pokročilejší grafické funkce. Dále pak zachytává vstup klávesnice a myši.

Třída také obsahuje jednoduchou grafickou konzoli, do níž lze psát textové příkazy. Viditelnost konzole je přepínatelná klávesou '~'. Dále také umožňuje vypisovat textové zprávy na obraz, jenž jsou ovlivněny viditelností konzole, pokud je vypnuta, zprávy se objeví jen na krátkou chvíli, jinak jsou zobrazeny všechny zprávy jenž se vejdou na obraz. Konzoli lze použít například pro ovládání klienta, posílání příkazů serveru a výpis textových zpráv uživateli. Mohou to být chybové či oznamovací zprávy.

3.7 Implementace komunikace mezi klienty a serverem

Tato kapitola popisuje implementaci práce s knihovnou ENet

3.7.1 Tvorba packetů

Pro konstrukci paketů na aplikační vrstvě byla napsána vlastní třída `TPack`. Tato třída je implementována přímo pro potřeby vkládání a čtení dat na jednom místě v paměti pro jednodušší práci s daty určenými ke komunikaci mezi aplikacemi. Po zavolání konstruktoru této třídy lze vložit libovolná data, řetězce různých délek, i bitové kopie objektů. Vše se poskládá za sebe do jednoho řetězce bytů a jde tedy přímo poslat po síti. Při čtení je opačný postup, pokud do konstruktoru v parametrech příjde ukazatel na data a délka dat, třídu lze využít pro čtení těchto dat stejným způsobem, jako funguje zápis.

3.7.2 Implementace komunikace pomocí ENet

Práce s ENet je docela jednoduchá, v prvé řadě je třeba připojit server na port, na kterém bude následně čekat příchozí klienty. Toho lze dosáhnout následovně:

```
ENetAddress address;
address.host = ENET_HOST_ANY;
address.port = port;
enet_initialize();
ENetHost* host = enet_host_create (& address, max_clients, channels, 0, 0);
```

Funkce `enet_host_create()` má pět parametrů, jsou to adresa, maximální počet připojených klientů a počet kanálů spojení, následovaný omezením stahovací rychlosti a odesílací rychlosti (v bytech za sekundu, 0 znamená neomezeno).

Klient pro připojení k serveru musí použít stejnou funkci inicializace, plus k tomu další, jenž jej spojí se serverem. `enet_host_connect` má čtyři parametry: spojovaný `ENetHost` host, adresa serveru s portem a počet kanálů k otevření. Dále pak uživatelská data, které zůstanou v tomto případě 0. Výstupem je ukazatel na `ENetPeer` obsahující data navázaného spojení. Ukázka kódu je znázorněna zde:

```
enet_initialize();
ENetHost* host = enet_host_create (NULL, 1, channels, 0, 0);
...
ENetPeer* peer = enet_host_connect (host, &server_address, channels, 0);
```

Tímto vznikne mezi serverem a klientem spojení. Dále je nutno zajistit získávání přijatých zpráv a událostí. K tomu slouží funkce `enet_host_service` použitá v následujícím příkladu. Jejím vstupem kromě `ENetHost` a `ENetEvent`, do nějž se uloží informace o události, je navíc čas v ms, který aplikace může čekat na příchod události, pokud žádná mezitím nenastala. V projektu je použita takto:

```
ENetEvent event;
while (enet_host_service(host, &event, time) > 0) {
  switch (event.type) {
    case ENET_EVENT_TYPE_CONNECT:
      // navázáno spojení
      ...
    case ENET_EVENT_TYPE_RECEIVE:
      // poslání paketu na přečtení
      ...
    case ENET_EVENT_TYPE_DISCONNECT:
      // spojení ztraceno
      ...
  }
}
```

Přijatá data jsou po příchodu k dispozici v `event.packet->data` jako ukazatel do paměti s délkou `event.packet->dataLength`.

Avšak důležitá část je i poslání dat. Funkce `enet_packet_create()` zaobaluje holá data o určené délce do paketu. Příznak `ENET_PACKET_FLAG_RELIABLE` dává funkci najevo, že pakety mají být doručeny v pořadí, jakém byly odesílány, pokud paket nedojde a ztratí se, je odeslán znovu. Následně je paket vyslán pomocí `enet_peer_send()` po spoji definovaným pomocí `peer`, tak jak je uvedeno zde:

```
void *data;
int len;
int channel;
//naplnění dat do data
...
ENetPacket *packet = enet_packet_create(data,len,ENET_PACKET_FLAG_RELIABLE);
enet_peer_send (peer, channel, packet);
```

3.8 Přenositelnost

Herní server byl navržen a implementován tak, aby byl přenositelný na Unix-like systémy a OS Windows. Je aplikovatelný jak na real-time, tak na tahové hry, na žánru pak vůbec nezáleží. Využívá knihovny ENet, SDL a OpenGL, jenž jsou taktéž přenositelné mezi systémy.

Kapitola 4

Demonstrační hry

V této kapitole jsou popsány vytvořené demonstrační hry s využitím herního serveru. Jako ukázku funkčnosti byla zvolena hra gomoku jako reprezentace tahových her, dále pak Ping-Pong reprezentující real-time hry.

4.1 Gomoku

Gomoku bylo vybráno jako ukázka funkčnosti tahových her na vyvíjeném herním serveru právě pro jeho jednoduchost a efektivnost.

4.1.1 Popis hry Gomoku

Tato hra se u nás obvykle hraje v období zvané piškvorky. Gomoku má však mírně odlišná pravidla. Hra probíhá na poli o dané velikosti 15x15, navíc oproti piškvorkám nevyhrává hráč, jenž dosáhne více než pět polí v řadě, pro výhru potřebuje právě pět.

4.1.2 Implementace Gomoku

Funkční hra vznikla rozšířením potomků základních tříd serveru a klienta a přepracováním jejich virtuálních metod. U tříd `TControlServer` a `TControlClient` je jen malá úprava, aby vytvářel správnou třídu pro hru. Dále pak proběhly zásadní úpravy v potomcích tříd `TGameSystemServer` a `TGameSystemClient` přidáním dvourozměrného pole hrací plochy 15x15 a reakcí na zprávy, týkajících se gomoku (zprávy jsou popsány v tabulce 4.1).

Maximální počet připojených klientů na jednu hru nebyl omezen. Nově příchozí, jenž se do hry nevejdou, se hry neúčastní, ale mají možnost pozorovat průběh jako diváci.

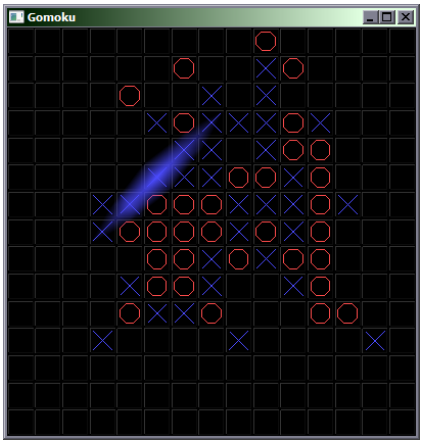
Zprávy klienta		
Typ	**význam**	**data**
0x01	Oznámení o kliku na pole	souřadnice x, y
Zprávy serveru		
Typ	**význam**	**data**
0x01	Kompletní hrací pole	data pole
0x02	Provedení tahu	souřadnice a barva tahu
0x03	Informace o výhře	souřadnice a vektor výherní řady

Tabulka 4.1: Seznam zpráv hry Gomoku

4.1.3 Implementace grafiky Gomoku a interakce s hráčem

Pro tuto část bylo použito knihoven SDL a OpenGL ve třídě `TWindow`. Po spuštění aplikace je vytvořeno okno pomocí SDL, které navíc obstarává vstup z klávesnice a myši. Gomoku využívá konzole třídy `TWindow` pro zadávání příkazů jako jsou přihlášení, připojení ke hře, výpis seznamu her atd., a ty pak posílá dál serveru na zpracování.

Po úspěšném připojení do hry se pak pomocí OpenGL zobrazuje hrací pole (názorně uvedeno na obrázku 4.1). Po kliknutí uživatele na některé pole je odeslána zpráva se souřadnicemi zamířeného pole na vyhodnocení serverem, ten pak zpětně posílá potvrzení tahu všem účastníkům hry.



Obrázek 4.1: Ukázka spuštěného klienta Gomoku

4.2 Ping-pong

Tato hra byla navrhnuta a vytvořena na ukázku a otestování funkčnosti pro real-time řízení času herního serveru.

4.2.1 Popis hry Ping-pong

Jedná se o jednoduchou hru odehrávající se na obdélníkovém hřišti, v níž je úkolem dostat míč do nepřítelovy brány pomocí myší řízeným kruhem. Hra není omezena jen na dva hráče, ale pro hratelnost je lépe hrát v sudém počtu, hráči se rozdělí do dvou týmů. Každý hráč dostane svůj kruh, jenž může řídit a narážet s ním do ostatních objektů.

4.2.2 Implementace hry

Základ implementace je obdobný jako u předchozí hry Gomoku. Nyní se však nejedná o tahovou hru, proto je tentokrát využito u potomka třídy herního systému funkce `loop`, jenž se bude volat periodicky 50 krát za sekundu. To zajistí docela plynulý chod simulace pohybu objektů po hracím poli.

Herní objekty

Všechny objekty v této hře jsou kruhové a definované jednou třídou. Míč a hráči se od sebe liší typem, velikostí, barvou, texturou a váhou. Objekty hráčů mají navíc nastavenu hodnotu určující vlastníka, jenž jej ovládá.

Herní objekty jsou definovány svou polohou, rychlostí a směrem, kterým jej hráč řídí. Jak server, tak klienti počítají fyziku objektů, klient při tom posílá serveru svůj směr pohybu, a server zpět klientům posílá stavy všech objektů. Tím, že klient počítá i sám fyziku na své straně, nemusí mu server posílat zprávy o objektech neustále každým cyklem, postačí jen při zásadnějších změnách či s větší periodou, což ušetří zbytečnou komunikaci. Klient je schopen s dostatečnou přesností předpokládat pohyby objektů i několik cyklů dopředu. Pro účely této hry bylo nastavena frekvence posílání zpráv na deset krát za sekundu, ukázalo se to jako vhodná volba. Klient však informuje server o svých pohybech neustále.

Nárazy objektů jsou definovány jednoduchými zákony fyziky, všechny objekty jsou kruhové, lze tedy jednoduše určit úhel dopadu a odrazu objektů. Detekce kolizí postupně zjišťuje kolize mezi všemi objekty a hranicemi hřiště.

Hráčův objekt nekopíruje přímo pohyby myši, nýbrž se jej snaží co nejlépe sledovat. Hráč má omezené zrychlení a rychlost, s jakou se může pohybovat. Jelikož zrychlení myši je okamžité, objekt je za kurzorem opožděn.

Góly hlídá pouze server, ten v případě gólu o něm informuje klienty. Tým dává gól, pokud se míč dotkne levé či pravé strany hřiště, kde je zvýrazněná oblast brány. To způsobí přidání bodu jednomu týmu a vyhazování míče uprostřed hřiště s krátkou přestávkou na zorientování hráčů. Server o celém dění na hřišti postupně informuje všechny hráče.

Zprávy mezi klientem a serverem

V tabulce 4.2 jsou zobrazeny potřebné zprávy posílané hrou Ping-pong. Klient vyvolává pouze dvě různé zprávy, jednu při pohybu myší nad hracím polem a druhou změnou týmu, tím dává najevo serveru svůj nový tým, server následně pošle aktualizaci ostatním klientům ve hře.

Z pohledu serveru jsou posílány 3 druhy zpráv. Pokud jsou nějak zásadně změněna herní data, například proběhl restart hry či změna týmu, posílá server klientům celkové obnovení dat. Další zprávu s obnovením dat objektů, jejich rychlostí a pozic, posílá server periodicky. A následující druh zprávy obsahuje informace o stavu gólů a čekacích dobách na míč při rozehrávce.

Zprávy klienta		
Typ	**význam**	**data**
0x01	Oznámení o pohybu	souřadnice x, y
0x02	Oznámení o změně týmu	tým
Zprávy serveru		
Typ	**význam**	**data**
0x01	celkové obnovení objektů	počet a data objektů
0x02	obnovení objektů	pozice, rychlosti a směry objektů
0x03	data stavu hry	góly a časy

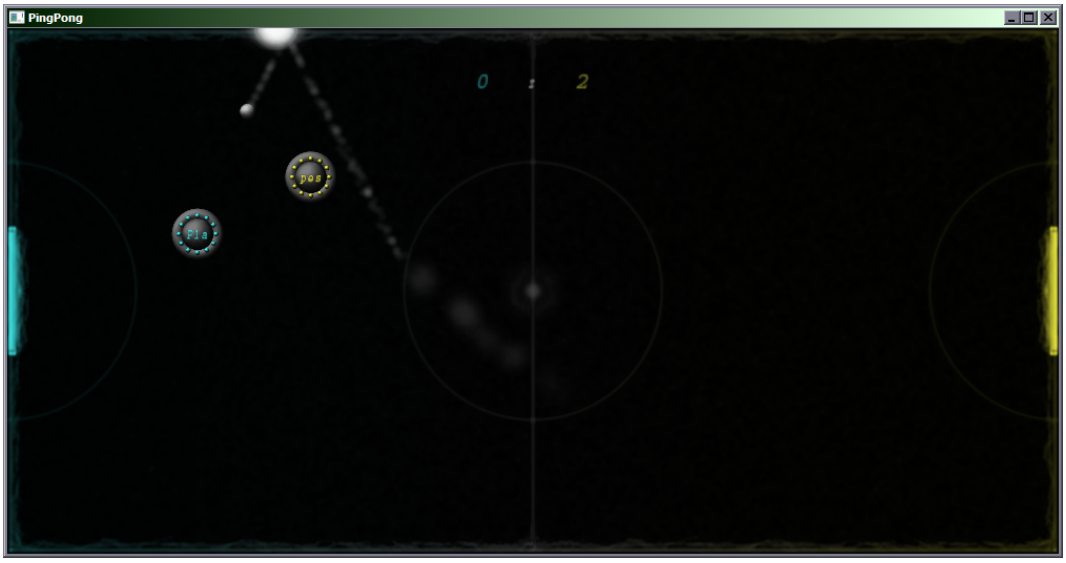
Tabulka 4.2: Seznam zpráv hry Ping-Pong

4.2.3 Implementace grafiky Ping-Pong a interakce s hráčem

Obdobně jako u gomoku byla implementována grafika a rozhraní této hry. Pouze zde přibylo použití textur navíc i s alfa kanálem (objekty jsou kruhové, ale textury čtvercové, proto jsou rohy textur průsvitné). Vykreslování probíhá cyklicky ve smyčce spolu s výpočtem fyziky objektů, takže bude probíhat s frekvencí 50 fps. Vykreslují se všechny objekty, každý má svou barvu, použité textury a velikost. Aplikace zaznamenává v průběhu hry pozice myši, a podle nich směruje hráčem ovládaný objekt. Tyto souřadnice zároveň posílá serveru.

Dalším grafickým prvkem oproti Gomoku je přidání jednoduchého částicového efektu za rychle pohybujícími se tělesy. Efekt je čistě grafický, server se jím tedy nezabývá a objevuje se pouze u klientů (názorná ukázka vzhledu je na obrázku 4.2).

Jednotlivé částice jsou generovány za objektem po překročení určité rychlosti, s dalším zvyšováním rychlosti přibývají i další částice. Jejich velikost je úměrná velikosti generujícího objektu, jejich životnost zůstává konstantní.



Obrázek 4.2: Ukázka spuštěného klienta Ping-Pong

Kapitola 5

Testování

Důležitou kapitolou je právě testování, nelze vydávat žádné hry, aniž by byly předem otestovány. Zvláště pak u her určených pro provoz po síti. Obě demonstrační hry byly otestovány v reálném provozu s reálnými hráči. Jako příklad je v následující kapitole popsán jednoduchý test demonstrační hry Gomoku.

5.1 Průběh testování Gomoku

Pro testování bylo využito výpisů přijatých a odeslaných zpráv herního serveru a reakcí aplikační hry postavené na serveru. Dva klienti byli spuštěni na dvou různých systémech (jeden na Windows Vista 32bit a druhý ve virtuálním Ubuntu 10.04 32bit) a serverová aplikace běžela na školním serveru Merlin.

Průběh testu byl následující. Nejdřív se připojil první klient, přihlásil se pod jménem "PlayerA"a založil hru. Druhý klient byl spuštěn, přihlásil se pod přezdívkou "PL2". Hráči odehráli několik tahů a následně bylo u prvního klienta cíleně vyvoláno odpojení od sítě. Následně byla síť znovu zapojena, hráč se znovu přihlásil a vstoupil do hry, po několika tazích server oznámil výhru. Oba hráči se odpojili a hra byla automaticky ukončena.

Výpis ze serveru vypadal takto:

```
connected (IP:port)
received  (IP:port) [Typ zprávy -délka dat zprávy]
send to   (IP:port) [Typ zprávy -délka dat zprávy]
```

Připojení hráče a vytvoření hry.

```
connected    (c2c0e593:53482).
received     (c2c0e593:53482) [c0 -11]      login "PlayerA"
send to      (c2c0e593:53482) [c0 -19]      stav přihlášení hráče OK
received     (c2c0e593:53482) [c4 -15]      nová hra
send to      (c2c0e593:53482) [c6 -4]       potvrzení vstup do hry
send to      (c2c0e593:53482) [cc -29]      textová zpráva o vstupu
send to      (c2c0e593:53482) [01 -231]     herní pole
```

Připojení druhého hráče do vytvořené hry.

```
connected    (c2c0e593:52413).
received     (c2c0e593:52413) [ca -0]       požadavek na seznam her
```

```
send to      (c2c0e593:52413) [c1 -0]       chybová hláška, požaduje login
received     (c2c0e593:52413) [c0 -7]       login "PL2"
send to      (c2c0e593:52413) [c0 -15]      stav přihlášení hráče OK
received     (c2c0e593:52413) [ca -0]       požadavek na seznam her
send to      (c2c0e593:52413) [ca -35]      informace: 1 běžící hra, 1 hráč
received     (c2c0e593:52413) [c6 -4]       požadavek na vstup do hry
send to      (c2c0e593:52413) [c6 -4]       potvrzení vstup do hry
send to      (c2c0e593:53482) [cc -25]      textová zpráva oběma účastníkům
send to      (c2c0e593:52413) [cc -25]
send to      (c2c0e593:52413) [01 -231]     herní pole
```

Poslání textové zprávy.

```
received     (c2c0e593:52413) [cc -11]      textová správa od "PL2"
send to      (c2c0e593:53482) [cc -16]      přeposlána všem účastníkům hry
send to      (c2c0e593:52413) [cc -16]
```

Tah hráče.

```
received     (c2c0e593:53482) [01 -8]       tah "PlayerA"
send to      (c2c0e593:53482) [02 -9]       výsledek tahu poslán oběma hráčům
send to      (c2c0e593:52413) [02 -9]
```

Po výpadku sítě jednoho klienta se zbývající připojený klient pokusí táhnout a posílá požadavek na seznam hráčů.

```
disconnected (c2c0e593:53482).              pád sítě "PlayerA"
send to      (c2c0e593:52413) [cc -32]      zpráva "player disconnected: PlayerA"
received     (c2c0e593:52413) [01 -8]       pokus o tah "PL2", není však na tahu
received     (c2c0e593:52413) [cb -0]       žádost o seznam hráčů ve hře
send to      (c2c0e593:52413) [cb -11]      seznam hráčů: "PL2" je jediný hráč
connected    (c2c0e593:55421).
received     (c2c0e593:55421) [c0 -11]      opětovné připojení hráče "PlayerA"
send to      (c2c0e593:55421) [c0 -19]
received     (c2c0e593:55421) [c6 -4]
send to      (c2c0e593:55421) [c6 -4]
send to      (c2c0e593:55421) [cc -29]
send to      (c2c0e593:52413) [cc -29]
send to      (c2c0e593:55421) [01 -231]
```

Výherní tah a konec hry.

```
received     (c2c0e593:55421) [01 -8]       výherní tah "PlayerA"
send to      (c2c0e593:52413) [02 -9]
send to      (c2c0e593:55421) [02 -9]
send to      (c2c0e593:52413) [03 -4]        oznámení o výhře
send to      (c2c0e593:55421) [03 -4]
received     (c2c0e593:55421) [c7 -0]       "PlayerA" opouští hru
send to      (c2c0e593:52413) [cc -32]       textová zpráva o odhlášení
send to      (c2c0e593:55421) [c0 -19]
received     (c2c0e593:55421) [c1 -0]       "PlayerA" se odhlašuje
send to      (c2c0e593:55421) [c1 -0]
disconnected (c2c0e593:55421).
```

Test dopadl úspěšně, po odpojení obou hráčů se hra automaticky ukončila a uvolnily se alokované místa v paměti. Test byl proveden pomocí programu valgrind, který nenašel žádné ztracené bloky dat.

5.2 Průběh testování multiplatformnosti

V průběhu testování mezi různými systémy však nastaly komplikace. Při pokusech přeposílání bitových kopií objektů mezi různými platformami vznikaly problémy při zarovnání dat objektů do paměti. Objekty na OS Windows měly jinou velikost a pozice dat ve strukturách oproti Linuxu. To však lze řešit přidáním `__attribute((packed))` k definici objektů. Výsledkem je minimalizace velikosti objektů, jejich data budou řazena tak, jak jsou popsána v definici přímo za sebou bez vynechaných bytů.

Další testování probíhalo úspěšně na systémech Windows Vista 32bit, Windows 7 64bit, Ubuntu 10.04 32bit, CentOS 5.4 64bit.

5.3 Bezdrátová komunikace

Další test proběhl za účasti dvou notebooků připojených k bezdrátové síti Wi-Fi a serveru spuštěného na školním serveru Merlin. Komunikace byla úspěšně navázána, podle očekávání spojení pomocí Wi-Fi nemělo znatelný vliv na hratelnost.

Kapitola 6

Závěr

V této práci byly popsány on-line hry více hráčů. Jejich možné rozdělení, výhody a nevýhody používaných technologií. Dále byl vysvětlen pojem MMOG či dedikované servery. Práce pojednává také o možnostech implementace herního prostředí a řízení herního času.

Výsledkem této bakalářské práce je pak návrh řešení a konstrukce herního serveru, schopného reagovat na požadavky klientů, a konstrukci klientů samotných. Implementovaný server vytváří prostor pro herní klienty a zprostředkovává spojení mezi hrou a těmito klienty. Klienti naopak zprostředkovávají interakci herního systému s koncovým uživatelem.

Konkrétní využití navrženého herního serveru je předvedeno na implementaci dvou demonstračních her. První hra je založena na principu tahových her, druhá hra se věnuje funkčnosti pro real-time řízení času. Obě hry byly testovány v reálném provozu a jsou plně funkční.

Herní server tedy bude použitelný i pro řadu jiných her, implementace byla provedena s ohledem na obecnost a univerzálnost, server je také přenositelný na různé operační systémy, testování probíhalo na OS Windows a Linux.

6.1 Možnosti rozšíření

Navržené řešení herního serveru lze dále rozvíjet, především v oblasti zabezpečení. Implementovanému hernímu serveru zatím chybí autentizace a autorizace hráčů, například pomocí SQL databáze a nějakého šifrovacího algoritmu pro znepřístupnění hesel hráčů. Tím by nabyla možnost i úschova dalších dat hráčů do databáze.

Další rozšíření vychází z faktu, že herní server byl implementován jako jednovláknová aplikace. Je však možné jej upravit tak, aby pracoval ve více vláknech. Obsluha her by pak neprobíhala postupně za sebou, ale mohly by být prováděna paralelně. Na vícejádrových procesorech by toto řešení mohlo mnohonásobně zvýšit výkon serveru.

Další možná rozšíření mohou být moduly zaměřené na specifický žánr, řízení objektů a třídy různých typů a podobně.

Literatura

[1] Machinarium [online]. 2009.
URL http://machinarium.net/demo/

[2] Minecraft [online]. 2011.
URL http://www.minecraft.net/

[3] RuneScape [online]. 2011.
URL http://www.runescape.com/

[4] Total War [online]. 2011.
URL http://www.totalwar.com/

[5] World of Warcraft [online]. 2011.
URL http://eu.battle.net/wow/en/

[6] The C++ Resources Network [online]. 2011 [cit 2011-4-21].
URL http://www.cplusplus.com/

[7] Blatný, M.: Dune 2 - pramáti všech strategií [online]. 2000.
URL http:
//doupe.zive.cz/pc/dune-2--pramati-vsech-strategii/sr-1-sc-112-a-17608

[8] Miller, T.: *Programujeme 3D Hry v jazyce C#*. Brno: Computer Press, první vydání, 2006, ISBN 80-251-1126-1, 335 s.

[9] Postel, J.: User Datagram Protocol. RFC 768, ISI, August 1980 [cit 2011-4-21].

[10] Postel, J.: Transmission Control Protocol. RFC 793, ISI, September 1981 [cit 2011-4-21].

[11] Salzman, L.: ENet [online]. 2011-2-9 [cit 2011-4-21].
URL http://enet.bespin.org

[12] Turek, M.: Seriál SDL: Hry nejen pro Linux [online]. 2005 [cit 2011-4-21].
URL http://www.root.cz/serialy/sdl-hry-nejen-pro-linux/

[13] Turek, M.: NeHe OpenGL Tutoriály [online]. 2008 [cit 2011-4-21].
URL http://nehe.ceske-hry.cz/tut_obsah.php

Příloha A

Obsah DVD

K práci je přiloženo DVD s výsledky této bakalářské práce. Obsahuje zdrojový kód technické zprávy v *latexu* a jeho přeloženou verzi ve formátu pdf. Se zdrojovými kódy jsou přiloženy i obrázky použité v práci jak v PostScriptu, tak v bmp či png formátech. Dále obsahuje zdrojové kódy herního serveru a klienta a zdrojové kódy demonstračních her. A také spustitelné binární soubory těchto her přeložené na systémech OS Windows Vista 32bit a CentOS 5.4 64bit Linux.