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

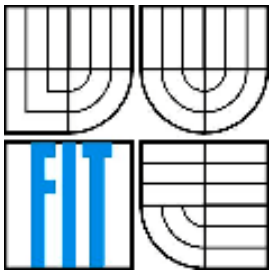
POČÍTAČOVÁ HRA PRO VÍCE HRÁČŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE MICHAL KUDR
AUTHOR

BRNO 2008

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

POČÍTAČOVÁ HRA PRO VÍCE HRÁČŮ

MULTIPLAYER COMPUTER GAME

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL KUDR

VEDOUCÍ PRÁCE
SUPERVISOR

ING. MICHAL HRADIŠ

BRNO 2008

Abstrakt

Cílem mojí práce je seznámení se základy tvorby počítačových her. V této práci je možné nalézt například základní informace o tvorbě herní scény, změnách stavu hry pomocí událostí z uživatelského vstupu a jednoduché řešení hry pro více hráčů. Pomocí získaných znalostí navrhuji jednoduchý tankový simulátor pro více hráčů v síťovém virtuálním prostředí.

Klíčová slova

FPS, herní scéna, objekt, textura, sprite, vybírání, pohyb, událost, protokol, klient-server, síťové virtuální prostředí

Abstract

My work objective is introduce game programming. It is able to find basic informations about game scene creating, game state changing by user input's events and simple resolution of multiplayer game in this work. By means of getting knowledge I propose simply multiplayer tank simulator in virtual network environment.

Keywords

FPS, game scene, object, texture, sprite, picking, movement, event, protocol, client-server, network virtual environment

Citace

Michal Kudr: Počítačová hra pro více hráčů, bakalářská práce, Brno, FIT VUT v Brně, 2008

Počítačová hra pro více hráčů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michala Hradiše
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Michal Kudr
14.5. 2008

Poděkování

Poděkování patří vedoucímu bakalářské práce Ing. Michalu Hradišovi za veškeré rady a nápady, které mi poskytl.

Obsah

1 Úvod

Počítačové hry pro více hráčů se staly fenoménem dnešní doby. Hraní tzv. multiplayerových her je velice populární zábava snad všech věkových kategrií.

Informační technologie je jedna z nejrychleji se odvíjejících oblastí moderní vědy. Platí to i pro rozvoj počítačové grafiky a síťové komunikace, které jsou nedílnou součástí tvorby multiplayerových her. Vývoj v této oblasti je tak rychlý, že hra vytvořená před rokem se v očích hráčů stává doslova „zastaralou“.

Před takovými patnácti lety byla tvorba počítačové hry relativně snadná. Hry vyráběli většinou jednotlivci a po několika dnech nebo týdnech práce byla hra hotová. Nicméně vypadala naprosto jinak než dnes. V tehdejších dobách se hrály hry, které měly pár desítek nebo stovek kilobytů a ukládaly se na diskety. Hry byly zkrátka „v plenkách“. Dnes je situace odlišná. Hry se neustále zdokonalují a jsou komplikovanější. Snaží se hráčům nabídnout krásnou a realistickou trojrozměrnou grafiku, dramatickou hudbu a zvukové efekty, a mnoho dalších prvků, jako je poutavý příběh, solidní umělá inteligence nebo třeba hrátky s fyzikou. Současné špičkové hry vyvíjejí velká studia s desítkami a stovkami programátorů, grafiků, designérů a dalších zaměstnanců. Dnešní hry jsou již dokonce tak složité, že je jejich tvorba často náročnější než produkce celovečerních filmů. Jejich tvorba trvá několik let a vývoj stojí i desítky milionů dolarů.

Práce obsahuje teoretický rozbor a jednoduchý postup pro vytvoření síťového virtuálního prostředí.

2 Čas a animační systém

2.1 FPS

Důležitým pojmem při programování her je FPS (Frames per Second)**.** Udává, kolikrát je za sekundu vykreslen obrázek hry (snímek neboli Frame). Pokud je FPS nízké, hra nebude působit plynulým dojmem.

Jaká hodnota FPS je správná? Její spodní hranici určuje lidské oko a kritická frekvence blikání (critical flicker frequency - CFF), což je hranice rychlosti blikání světla, při jejímž dosažení se toto světlo jeví, jako by svítilo nepřerušovaně. K tomu dochází, v závislosti na intenzitě světla, někde mezi 10-ti a 50-ti Hz (10-50 FPS).Přibližná hodnota FPS, která zajistí plynulost se pohybuje kolem čísla 25. Horní hranice FPS je dána obnovovací frekvencí monitoru. Ta je obvykle mezi 70 a 90 Hz (70 až 90 FPS). Nemá smysl, aby program posílal grafické kartě více snímků za sekundu, protože přebytečné snímky se nezobrazí. Příliš vysoká hodnota FPS spotřebuje zbytečně velké množství procesorového času a zatěžuje grafickou kartu.

S pojmem FPS souvisí problém různé výkonosti grafických karet, z čehož vyplývá, že hodnota FPS bude u stejné hry různá v závisloti na použité grafické kartě. Při programování her je nutné zajistit,aby hra běžela stejně rychle na různých počítačích. K vyřešení tohoto problému se dají použít dva základní funkční způsoby.

Konstantní čas

Jedná se o jednodušší řešení. Na začátku se zvolí pevná doba, která určí, jak často se bude měnit stav hry. Například pro nejnižší možný počet snímků za sekundu (FPS = 25) bude doba mezi jednotlivými snímky rovna hodnotě 40 ms. Vytvoříme funkci, která bude hlídat čas okolo 40 ms. Protože se občas musí počkat na překreslení obrazovky nebo na plánování OS, bude měření času trochu nepřesné. Může se tedy stát, že hra poběží na různých počítačích s nepatrně odlišnou rychlostí.

Proměnný čas

Tato metoda je výrazně přesnější z hlediska přesnosti času, ovšem její implementace je o to pracnější. Pricip metody spočívá, v tom, že funkci sloužící k pohybu hry řekneme dobu od posledního volání a ta se postará o přesnou aktualizaci. Funkci pro pohyb lze napsat a používat jako diskrétní simulaci. Různé objekty ve hře lze psát s pevným časem na změnu a navíc může mít každý objekt tento pevný čas jiný. Diskrétní simulace funguje tak, že si udržuje frontu událostí tříděnou podle času. U každé události je čas,

za jak dlouho se má provádìt. Když se řekne simulaci, že uběhlo 25ms, podívá se do fronty a provede všechny události, které se mají stát v následujících 25ms.

Konstantní čas	Proměnný čas
Přibližná rychlost	Přesná rychlost
Na pomalých počítačích pomalé	Na pomalých počítačích trhané
Užívá jen tolik času CPU, kolik potřebuje	Užívá všechen dostupný čas CPU
Na rychlejším počítači beze změn	Na rychlejším počítači více FPS
Jednodušší funkce na pohyb	Komplikovanější funkce na pohyb

Obrázek 1. Srovnání konstatntního a proměnného času

3 Grafika

Při tvorbě her je grafika velice důležitá věc, ne-li nejdůležitější. V současné době je velice moderní 3D grafika a 3D hry. Trojrozměrná grafika nachází v současnosti kromě počítačových her uplatnění v mnoha oborech, mezi jinými ve filmové tvorbě a televizní produkci. K dvourozměrným bitmapám a vektorům se zde přidává třetí prostor a vytvořeným 3D objektům se přiřazuje celá řada zobrazitelných vlastností. Základem objektů ve třetím rozměru jsou tzv. drátěné modely, jež vytvoří tvarovou podobu obrazu a posléze jsou na jednotlivé plochy přidány tzv. textury (zobrazitelné vlastnosti povrchu tělesa nebo plochy).

3.1 OpenGL

OpenGL (Open Graphics Library) je nízkoúrovňová knihovna pro práci s trojrozměrnou grafikou. Od doby svého uvedení na počátku devadesátých let se stala široce uznávaným a podporovaným standardem na poli grafických aplikací, CAD/CAM/CAE inženýrských systémů, virtuální reality a tvorby her. OpenGL představuje jednotné API (Application Programming Interface) mezi programem a grafickým hardware. Hlavními rysem OpenGL je nezávislost na cílové platformě a použitém programovacím jazyku. Více o OpenGL v [2].

3.2 DirectX

Microsoft DirectX je programátorská knihovna obsahující nástroje pro tvorbu počítačových her a dalších multimediálních aplikací, vytvořená firmou Microsoft pro použití pod operačním systémem Windows. Aktuální verzí knihovny je momentálně DirectX 10.

DirectDraw - ve verzi 8.0 DirectDraw zaniklo a veškerá 2D grafika se dělá pomocí Direct3D
Direct3D
DirectSound - i DirectSound je ve verzi 8.0 pod DirectMusic
DirectMusic
DirectInput
DirectPlay - podpora hry více hráčů po síti
DirectSetup – jednoduchý nástroj umožňující instalaci knihovny DirectX na počítač,
DirectX Media Objects

Direct3D

Základní principy jsou podobné jako v DirectDraw, ale kromě toho umožňuje modelování objektů v prostoru tzn. že je 3D. V dnešní době většina komerčně úspěšných her používá rozhraní Direct3D. U dřívějších verzí D3D si vývojáři stěžovali na složitost a neefektivnost kódu oproti OpenGL, které bylo mnohem jendodušší a tudíž i rychlejší. Ve verzi 8.0 se však vývojaři Microsoftu plácli přes ruce a kompletně D3D přepsali tak, že se minimálně vyrovná OpenGL. Ovšem narozdíl od OpenGL se Direct3D neustále vyvijí a prakticky každá verze SDK přinese něco nového. Takže D3D mnohem lépe dokáže využít dnešní hardware co se týče technologie.

DirectSound

Jak název napovídá, DirectSound zabývá oblastí zvuku neboli jak relativně jednoduše implementovat do vašeho programu zvukovou kulisu. Jde to opravdu jednoduše. Od verze 8.0 je lepší využít DirectMusic, které je ještě jednodušší a pro základní ozvučení postačující.

DirectMusic

Tato komponenta přibyla do DirectX ve verzi 7.0, kde ovšem měla malou podporu. Až ve verzi 8.0 je schopná být opravdu užitečná. Můžete s ní přehrávat hudbu i zvuky, prostě vše na co si vzpomenete. Podpora EAX (Enviromental Audio Extension) a 3D zvuku je samozřejmostí.

DirectInput

Je součást, která pomáhá používat ve vaší aplikaci vstupní zařízení tzn. klávesnice, myš nebo třeba joystick.

DirectPlay

Poslední část je pro spojení vaší aplikace s jiným počítačem nebo s Internetem (pomocí protokolu TCP/IP).

4 3D počítačové hry

4.1 Osvětlení herní scény

Kapitola obsahuje základní informace o osvětlování scény. Více v [13].

Ambientní světlo

Přichází ze všech směrů se stejnou intenzitou. Nahrazuje ve scéně tu část reálného světla, která obyčejně vzniká odrážením světla od různých předmětů, například nábytku a stěn v místnosti.

Směrové světlo

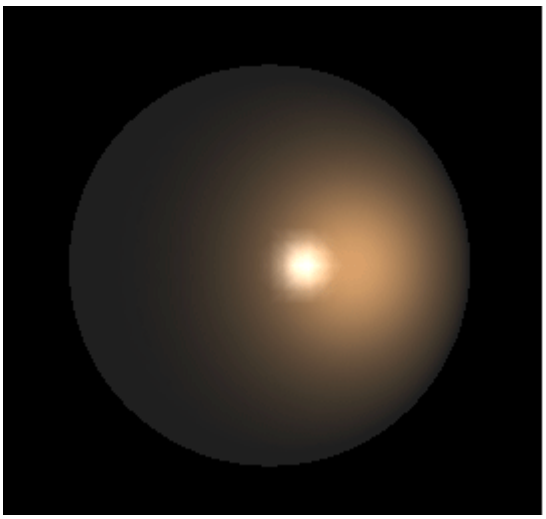
Směr a intenzita jsou ve všech bodech scény stejné. Toto světlo odpovídá světlu přicházejícímu z nějakého vzdáleného zdroje, například Slunce. Světlo se konstruuje pomocí barvy světla a vektoru určujícího jeho směr.

Bodové světlo

Je charakterizováno svou barvou, pozicí ve scéně a třemi koeficienty (konstatntí, lineární, kvadratický), které určují, jak se se vzdáleností mění jeho intenzita.

Reflektorové světlo

Je podobné bodovému světlu v tom, že se také šíří z jednoho bodu a se vzdáleností od zdroje se může zeslabovat, nicméně se na rozdíl od něj nešíří do všech směrů, ale pouze do zadaného kužele.



Obrázek 2. Koule osvětlená reflektorovým a ambientním světlem

4.2 Textury a texturování

Textury jsou nezbytná součást jakékoliv počítačové hry. Umožňují větší realističnost a jednodušší geometrii modelů. Principem texturování je obarvení povrchu zobrazovacích těles různými obrazci. Důležité je, že se nijak nemění geometrické vlastnosti těles, pouze se jinak zobrazuje jejich povrch. Obrazce, které se na povrch těles nanášejí, se nazývají *textury* (textures). Tyto textury jsou většinou představovány plošnými obrázky (dvoudimenzionální textury), některé grafické systémy však podporují i vykreslování jednorozměrných a dokonce trojrozměrných (objemových) textur. Více o texturách lze nalést v [4].

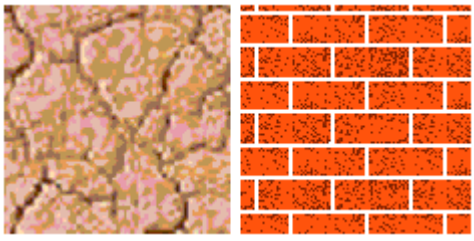
Textury dělíme:

1) podle dimenze
   a) 1D
   b) 2D
   c) 3D

2) podle reprezentace
   a) Procedura
   b) Mapa - Používají se nejčastěji. Texturou je bitmapový obrázek. Důležité je rozlišení obrazu, a tím i dostatečná detailnost textury. Právě tyto textury se používají při tvorbě her.

Použití textur:

1) vytvoření textury
   a) načtení mapy
   b) generování funkcí (procedurou)
2) nanesení textury - hledání vhodné projekční funkce



Obrázek 3. Ukázky textur vytvořené pomocí bitmapového obrázku

Podpora textur v OpenGL

V OpenGL jsou podporovány pouze rastrové textury, které mohou být buď jednodimenzionální, dvoudimenzionální, nebo třídimenzionální. V případě *jednodimenzionálních textur* se prakticky jedná

o pruh pixelů, pomocí něhož se dají realizovat různé barevné přechody. *Dvoudimenzionální textury* (rastrové bitmapy a pixmapy) jsou v dnešní době zdaleka nejpoužívanější a jsou podporovány na naprosté většině grafických akcelerátorů.

4.3 3D modely

Složité geomterie můžeme sestavovat přímo v aplikaci. Sami však musíme provést výpočet 3D souřadnic a zajistit jejich správné řazení unitř geometrických objektů, což se dá poměrně dobře zvládnout jen u nejjednodušších tvarů, jako jsou například krychle nebo kužely. Mnohem rozumnější a efektivnější způsob je vytvořit objekt pomocí 3D-modelovacího softwarového nástroje a poté jej do aplikace načíst (externí modely).

4.4 Náhodná tvorba herní scény

V praxi existují případy, kdy potřebujeme, aby se herní scéna vytvořila automaticky (rozmístění hráčů, překážek či sklon krajiny atd.). K tomu se používají generátory náhodných čísel, kterými při například umisťování objektů čí překážek do scény lze vygenerovat jejich náhodné souřadnice nebo jejich počet atd. Tímto způsobem se docílí toho, že při spuštění hry se vygeneruje vždy naprosto odlišná krajina.

Alternativou k tomuto postupu je sestrojení generátoru náhodných čísel, který generuje sekvenci náhodných čísle v závisloti na specifické hodnotě,tzv. semínku (seed). Pokud se generátoru zadá pokaždé stejné semínko, vygeneruje identickou řadu čísel, tzn. krajina bude při opakovaném spuštění vždy stejná. Tato vlastnost je užitečná ve hrách, kde potřebujeme krajinu zafixovat, jako například ve hrách pro více hráčů, kde se o generování scény stará klient. Při použití prvního uvedeného způsobu každý klient vygeneruje odlišnou krajinu, z čehož plyne, že hra by byla nehratelná. Použití generátoru a semínka tento problém odstraňuje. Pro vygenerování semínka se napřídlad dá použít libovolná metoda random(), popřípadě přečtení systémového času, čímž se docílí toho, že při založení nové hry bude krajina jiná.
Musí se zajistit, aby hodnota vygenerovaného semínka byla přístupná všem uživatelům. Jeho hodnota se například dá umístit na server, kde si ji po připojení každý klient přečte a na jeho základě vytvoří scénu. Výsledkem je totožná scéna u všech klientů.

4.5 Sprity

Aktivní herní entity mají často formu spritů. Sprite je pohybující se grafický objekt, který může reprezentovat hráče (reaguje tedy na stisky kláves nebo na aktivitu myši) nebo může být řízen nějakým „inteligentním“ podprogramem hry.

4.6 Vybírání objektů ve scéně

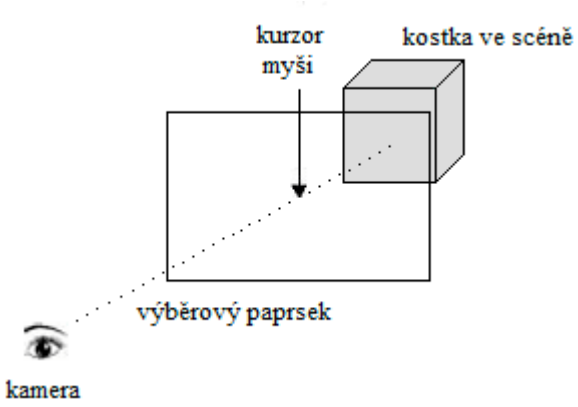
Vybírání objektů (picking) je technika hojně využívaná při tvorbě her.

Obecně se dá říct, že vybírání je uživatelsky příjemnějším způsobem pro získání vstupu od uživatele, než kdybychom se dotazovali například na souřadnice či název nějakého objektu.

Vybírání je v podstatě zvolení určitého tvaru (či tvarů) ve scéně, k čemuž obvykle dochází tak, že uživatel na tento tvar klepne kurzorem myši. Jeho implementace spočívá v projekci výběrového tvaru (čáry nebo paprsku) od pozorovatele do scény přes pozici kurzoru myši na obrazovce a ve výpočtu průniku s nejbližším objektem ve scéně.

Další možností je výběrový tvar nasměrovat jakýmkoliv směrem do scény z libovolného bodu. Například pohybující se objekt může vysílat při aktualizaci každého snímku více paprsků v libovolných směrech a testovat vzdálenost průsečíků. Tento způsob lze použít v počítačových hrách například ke sledování terénu a detekci kolizí.

Pro uvedený postup existuje řada variant, např. použití odlišného výběrového tvaru (kužele či válce) namísto čáry. Další možnotí je, aby výsledek vybírání obsahoval všechny protnuté objekty namísto jedinného, který se nachází nejblíže pozorovateli. V grafu scény odpovídá zvolený objekt listovému uzlu, které lze ve výchozím stavu vybírat, takže programátoři by měli vypnout vybírání u co největšího počtu uzlů, aby tak snížili cenu testování na průnik. Ve velkých herních scénách, jež obsahují tisíce viditelných objektů, se jedná o velmi důležitou optimalizaci.



Obrázek 4. Vybírání objektů na obrazovce pomocí kurzoru

5 Počítačové hry pro více hráčů

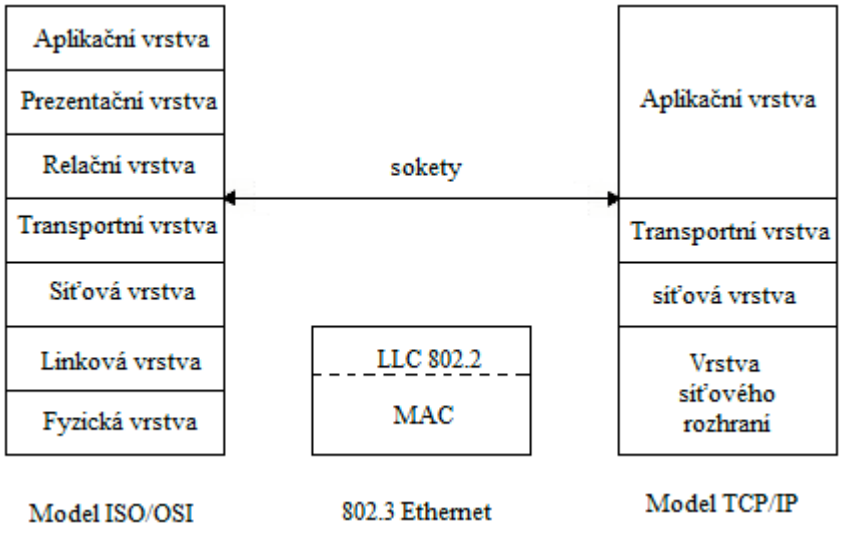
5.1 Základy síťové komunikace

Tako kapitola představuje jednoduchý úvod do problematiky počítačových sítí. Seznámíme se zde se základy, jejichž znalost je nezbytná pro tvorbu síťových her a síťových virtuálních prostředí. Více informací lze nalézt například v [10].

5.1.1 Komunikace mezi dvěma systémy – TCP/IP

Kapitola obsahuje stručný úvod o komunikaci a TCP/IP podle [5] a [6].

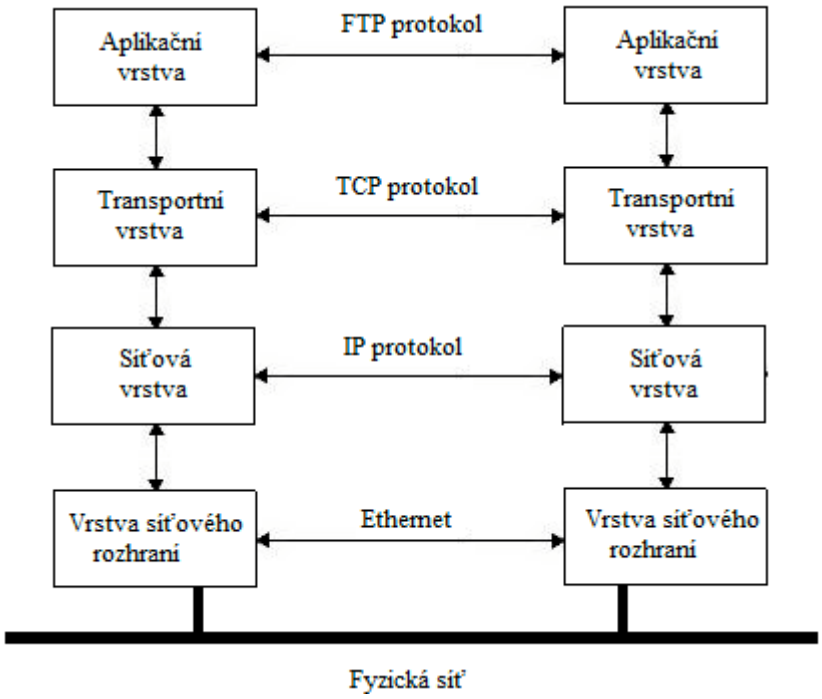
TCP/IP je souprava protokolů, původně vyvinutá na začátku osmdesátých let institucí DARPA. Její široká popularita pochází z toho, že byla implementována na všech strojích, osobními počítači počínaje a superpočítači konče, a že není závislá na konkrétní společnosti, a že se hodí pro všechny druhy sítí, od typu LAN až po internet. Oproti standartu ISO/OSI se TCP/IP skládá ze čtyř vrstev. Konkrétně vrstva síťového rozhraní, síťová vrstva (někdy uváděná jako internetová vrstva), transportní vrstva a aplikační vrstva.



Obrázek 5. Porovnání ISO/OSI a TCP/IP

V rámci konkrétní vrstvy to vypadá, jako by se data přenášela přímo do ekvivalentní vrstvy druhého systému. Ve skupečnosti se data předávají skrze vrstvy směrem dolů, přes fyzickou síť a nakonec směrem nahoru, přes všechny vrstvy druhého systému, viz. obrázek 6, převzatý z [1].

O doručování datagramů mezi koncovými transportnímy body jakýchkoli dvou strojů na internetu se stará transportní vrstva. Lokacce aplikace se specifikuje IP-adresou hostitele a číslem portu, na kterém „naslouchá“. Protokoly transportní vrstvy jsou UDP a TCP. Komunikace mezi dvěma systémy je znázorněna na obrázku 6 i s příklady používaných protokolů.



Obrázek 6. Komunikace mezi dvěma systémy

Protokol UDP realizuje nespojovanou transportní službu: zpráva uživatele se rozdělí na datagramy a každý datagram se do cíle pošle jednou z dostupných cest. Neexistuje zde žádný dlouhodobý dedikovaný spoj vytvořený mezi těmito dvěma systémy. Protokol UDP dědí nevýhody protokolu IP: datagramy mohou dorazit v jakémkoliv pořadí a jejich doručení není zaručené. Tento protokol je často přirovnáván ke službám klasické pozemní pošty.

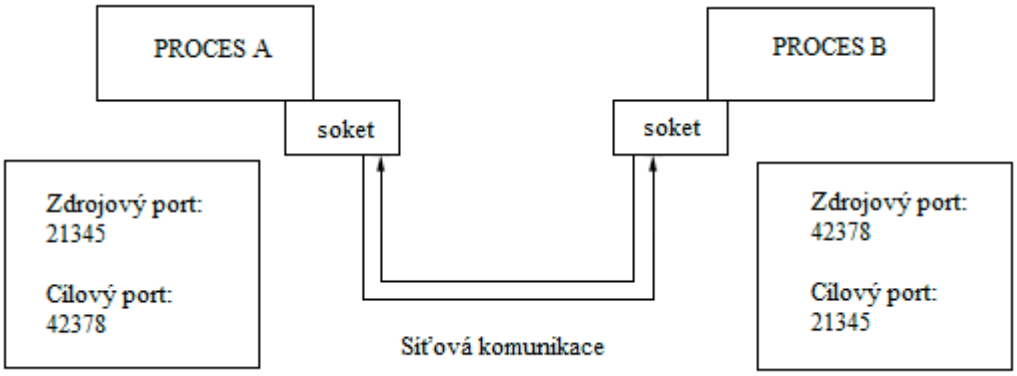
Protokol TCP je na druhou stranu transportní službou spojovanou. Z pohledu uživatele se jedná

o dlouhodobý, dedikovaný spoj ustavený mezi odesilatelem a příjemcem, na němž je možné provozovat obousměrnou proudovou komunikaci. Z tohoto důvodu je protokol TCP přirovnávám k telefonní službě.

Protokol TCP používá sofistikovanou interní kontrolu chyb, aby zajistil, že jednotlivé složky TCP-datagramů dorazí ve správném pořadí a že se žádný z nich neztratí. Zátěž způsobená tímto procesem může být v případě herních aplikací, jež si cení především nízkého zpoždění, příliš závažná.

5.1.2 Sokety

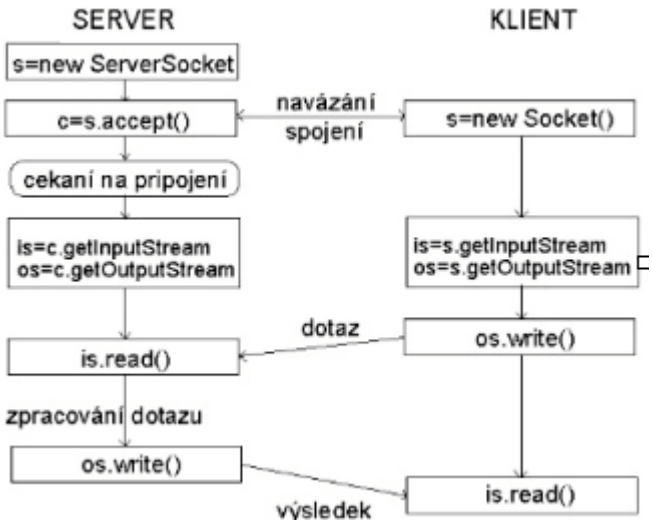
Socket je softwarová entita reprezentující zakončení spojení mezi dvěma stroji v síti. V podstatě se jedná o datovou strukturu umožňující pracovat s uzlem v síti podobně jako s lokálním souborem. Jsou jednoznačně identifikovatelné v síti – IP adresa a číslo portu.



Obrázek 7: Použití soketů

Posloupnost operací při spojení pomocí socketů na úrovni TCP

1) Server nastaví číslo portu, na kterém bude poslouchat a čeká na požadavky.
2) Klient se pokouší o spojení na dané adrese a daném čísle portu.
3) Pokud bylo spojení úspěšně navázáno, komunikace dále probíhá pomocí proudů



Obrázek 8: spojení pomocí soketů na úrovni TCP

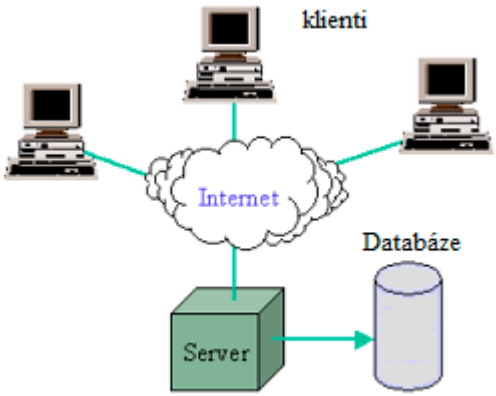
5.1.3 Model klient-server v počítačových hrách

Model klient-server je nejběžnější síťovou architekturou využívanou v distribuovaných aplikacích. Server je program (nebo kolekce spolupracujících programů), který poskytuje služby a nebo spravuje zdroje jiných programů, které tvoří jeho klienti.

Klíčovou výhodou modelu klient-server je schopnost řídit své klienty umístěním důležitých zpracování a dat do svých vlatních prostředků. Příkladem v on-line hrách může být rozhodování o výsledku hráčových akcí, jako například střelba na libovolný objekt. Další možností je pověřit touto činností klienta.

Soustředění větší části aplikace na serveru („tlustý“ server) umožňuje snažší údržbu a aktualizaci. Umístění stavových informací do jednoho místa umožňuje velice jednoduché změny a zabraňuje nekonzistencím, které vznikají při aktualizaci více kopií na straně klientů.

Nevýhodou takového serveru je jednak přetížení při připojení velkého množství klientů, při kterém naroste zpoždění na nepřijatelnou úroveň, ale i spolehlivost, a to v tom smyslu, že selhání serveru ovlivní všechny připojené klienty. Tento problém se dá řešit rozšířením počtu serverů, které se můžou specializovat na vykonání určitých požadavků klienta. Mohou také plnit tzv. roli „dvojníků“, kteří jsou připraveni nahradit server, který selhal. Dalším způsobem je přiřazení obsluhy různých lokací, kde obsluhují jen místní klienty. Při překročení hranic oblasti virtuálního světa dojde k přepojení na jiný server. Více informací o použití modelu klient-server v počítačových hrách lze v [1], kapitole o základech počítačových sítí.



Obrázek 9. Jednoduchá architektura klient-server

5.2 Síťová virtuální prostředí

Síťové virtuální prostředí (Networked virtual environment) je počítačový, uměle vytvořený svět 3D-prostorů, jehož návštěvníky jsou geograficky rozmístění uživatelé, kteří spolu navzájem komunikují a spolupracují.

Síťová virtuální prostředí jsou potomky her typu MUD (Multiple-User Dungeon), což jsou textové dobrodružné hry s hrdinou v hlavní roli, které se těší veliké popularitě již od sedmdesátých let minulého století. V devadesátých letech se začaly používat pro implementaci těchto světů objektově orientované techniky (MOO – MUD object oriented), díky kterým se zrodily první 2D- a 3D-diskusní prostředí.

Síťová virtuální prostředí jsou v současné době, kromě využití v počítačových hrách, předmětem intenzivního akademického výzkumu na celém světe. Více informací lze nalézt v [1].

5.2.1 Prostory

V síťovém virtuálním prostředí definují 3D-prostory topologii virtuálního světa. Svět může být tvořen jedinou velkou oblastí (např. krajina, ulice nebo hřiště), malým soukromým prostorem pro vybrané skupiny uživatelů (např. konferenční místnost, tělocvička nebo hala) nebo místem pro individuální interakci (například kancelář nebo kuchyně). Prostory mohou být buď neměnné nebo nataveny tak, aby jim uživatelé s odpovídajícími právy směli upravovat paramentry, mazat či vytvářet prostory nové.

5.2.2 Uživatelé

Uživatelé síťového virtuálního prostředí jsou vizuálně reprezentováni pomocí avatarů, vytvořených jimi samotnými při jejich prvním připojení do virtuálního světa. Na implementační úrovni může být uživatel realizován dvěma druhy avatarů – lokálním avatarem přítomným na hráčově stroji a stínovým avatarem používaným ve všech ostatních strojích připojených k virtuálnímu světu. I když oba typy avatarů vypadají na obrazovce stejně, v komunikační vrstvě se chovají scela odlišně. Lokálního avatara může uživatel ovládat přímo, bez zátěže plynoucí z komunikace přes server. Stínový avatar vyžaduje, aby se aktualizace jeho stavu doručovaly po síti, čímž vznikají problémy způsobené zpožděním.

5.2.3 Objekty

Objekty prostoru můžeme klasifikovat dvěma způsoby. Některé objekty, jako jsou budovy, ukazatele či součásti nějaké ulice, jsou neměnné. Jiné mohou být přenositelné, i když stále pasivní. Pohyb ve virtuálním světě může na implementační úrovni vyžadovat odpovídající pohyb reprezentace objektu mezi počítači. Objekty mohou při „spuštění“ reagovat – například dveře se při dotyku otevřou. Dynamický objekt másvoje vlastní chování, často na bázi pseudo-inteligence, což mu umožňuje pestřejší interakci s uživateli, kterou tak může zahájit i samotný objekt.

Objekty také představují jeden ze způsobů komunikace mezi uživateli, kteří si je předávají, vyrábějí pro druhé jejich kopie nebo je dělí na menší části.

5.2.4 Pohledy

Pohledy určují, jakým způsobem uvidí klient prostory, objekty a ostatní uživatele. Většina her je orientovaná na pohled z perspektivy hrající osoby, takže hráč ze svého avatara moc nevidí. Každému uživateli lze nabídnout možnost využití i několika dalších pohledů do prostoru, které pak může dle potřeby dynamicky upravovat.

6 Vlastní řešení

6.1 Vytvoření scény

Vvtvořené virtuální prostředí je složeno z podlahy, překážek ve formě krychlí a válců a jednotlivých hráčů ve formě spritů. Scéna je osvětlená ambientním světlem a je navíc pokrytá lineární mlhou. Více o vytváření herní scény lze nalézt v [1] nebo [7].

Při generování scény jsem využil jednoduchý generátor náhodných čísel popsaný v [12], který generuje posloupnost náhodných čísel na základě hodnoty semínka, vygenerovaného serverem a obdrženo při navázání spojení.

Server rovněž generuje hodnoty představující počet krychlí a kostek ve scéně. Vygenerovaná posloupnost čísel poslouží jak pro náhodné umístění překážek do herní scény, tak i pro nastavení míst, na kterých se může objevit nový uživatel po připojení do hry nebo po znovuumístění hráče do scény, pokud byl zasažen. Po vygenerování souřadnic probíhá ověření, zda se na nich nenachází překážka. Výsledkem je 10 míst ve scéně, na kterých je možné umístit uživatele, ze kterých je vždy náhodně jedno vybráno. Po vybrání lokace musí následovat ověření, zda se na něm nebo v jeho nejtěsnějším okolí nenachází jiný hráč. Pokud tato situace nastane, následuje náhodné vybrání nové lokace a na něj je umístěn sprite hráče.

6.2 Pohyb spritu

Pohyb hráče ve scéně je realizován pomocí zachytávání událostí z klávesnice.

Aplikace detekuje stisk, popřípadě uvolnění klíčových kláves. Pokud dojde k události stisku klávesy, následuje ověření, zda požadovaný pohyb neodporuje již probíhajícímu pohybu (pokud je stisknutá klávesa pro pohyb vpřed, detekce klávesy pro pohyb dozadu se ignoruje). V případě, že stisk klávesy platný, je tento stav zaznamenán spolu s časem stisknutí.

Pohyb se realizuje při každém snímku dotazem na stav jednotlivých kláves. Pokud je zjištěna stisknutá klávesa, výpočet velikosti posunu hráče.

Pro rotaci pohledu a věže tanku se zachytávají události pohybu myši. Při každém snímku se zjistí x-ová souřadnice kurzoru. Pro jednoduché určení směru rotace stačí dvě hodnoty, a to poslední a aktuální pozice kurzoru na obrazovce. Při detekci pohybu myši se aktuální pozice kurzoru nastaví jako poslední a nová pozice jako aktuální. Při každém snímku se porovnávají x-ové souřadnice a provede se požadovaný pohyb.

V praxi tento algoritmus může z důvodu porovnání pouze dvou posledních souřadnic způsobovat trhaný pohyb. Pro vyřešení problému lze uchovávat souřadnice kurzoru z více posledních snímků a pro porovnání zvolit aktuální a nejstarší dostupnou souřadnici.
Více o ovládání pomocí uživatelského vstupu v [7].

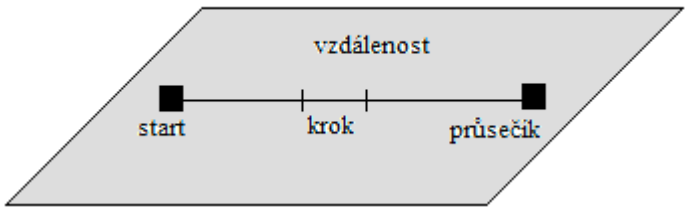
6.3 Střelba

6.3.1 Vystřelení náboje

Princip střelby spočívá v použití techniky vybírání objektů na obrazovce při stisknutí tlačítka myši. Jako souřadnici na obrazovce nejsou použité souřadnice kurzoru, ale souřadnice středu obrazovky, kde se nachází zaměřovač.

Pro jednoduchou reprezentaci střely jsem zvolil bílou kouli, která je před výstřelem neviditelná a umístěná na konci děla.

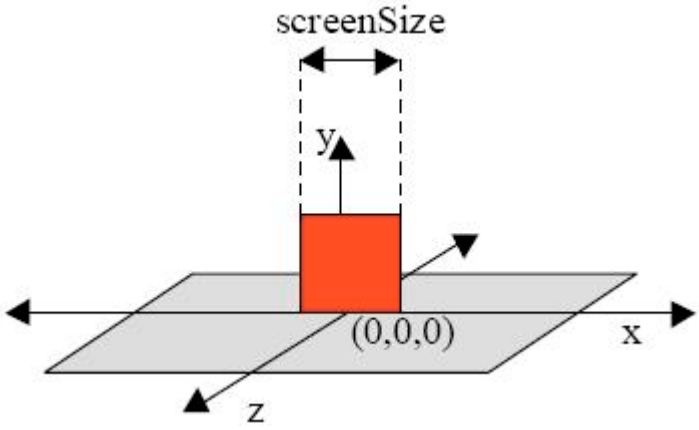
Po stisku tlačítka dojde k ověření, zda uplynul čas od posledního výstřelu určený k nabití (3 sec) a pokud ano, dojde k předání zprávy `username fire d` ostatním klientům. O výpočty dráhy letu a dopadu střely se stará už každý klient sám. V lokální scéně dojde k přehrání bodového zvuku výstřelu z aktulání pozice hráče, střela se zviditelní a pohybuje se po přímce od svého počátečního bodu do bodu, reprezentující průsečík ve scéně, zjištěný pomocí vybírání. Pohyb střely je realizován pomocí kroků, jejichž velikost se mění v závislosti na čase. Let střely je omezen vzdáleností počátečního bodu a průsečíku. Potenciálním problémem může být situace, kdy po posunu o vypočítaný krok dovolenou vzdálenost střela překročí. Pokud tato situace nastane, krok střely je upraven na hodnotu celková vzdálenost – uražená vzdálenost, čímž se docílí exploze přesně v místě dopadu. Jakmile střela urazí vypočítanou vzdálenost, dochází k explozi a střela se neviditelná vrátí zpět na pozici na konci děla.



Obrázek 10. Pohyb střely

6.3.2 Exploze

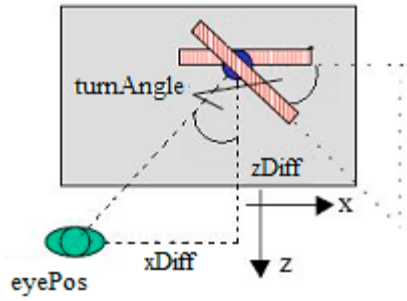
Visuální složka výbuchu je podle realizovaná jako série textur, které jsou vykreslované jedna přes druhou na povrch čtyřúhelníku, který je tvořen jedinným čtvercem o straně velikosti `screenSize`, jehož čelní stěna je natočena podél kladné osy Z. Předpokladem pro pro správné zobrazení je, že načítané obrázky budou čtvercového tvaru. V opačném případě by při jejich ukádaní na stěnu čtyřúhelníku došlo k zkreslení. Uzel uchovávající sekvenci obrázků je připojen k uzlu střely a při dopadu střely dojde ke zviditelnění a spuštění sekvence.



Obrázek 11. Čtyřúhelník pro zobrazování série textur

Problém nastává při dopadu střely a zobrazení exploze. Čtverec pro zobrazování sekvence obrázků výbuchu je natočen vůči poloze výstřelu, takže nesměřuje k výhledu lokálního uživatele.

Při řešení jsem vycházel z [7]. Situaci popisuje následující obrázek 12 převzatý z [11]. Šrafovaný obdelník reprezentuje čtyřúhelník exploze, který je původně natočen ve směru kladné osy Z. Lokální klient se nachází v bode `eyePos`, jenž je od čtyřúhelníku posunutý o `xDiff` dílků podél osy X a `zDiff` dílků podél osy Z. Úhel turnAngle se vypočítá jako arkustangens z `xDiff` a `zDiff`.



Obrázek 12. Natočení exploze směrem k pozorovateli

6.4 Detekce kolize

Pro řešení detekce kolizí je využívána technika vybírání, a to takovým způsobem, že při každém pokusu o pohyb, dojde k nasměrování výběrového tvaru, konkrétně válce o průměru rovnající se šířce tanku, směrem požadovaného pohybu. Pomocí zjištěného nejbližšího průsečíku se vypočítá vzdálenost mezi tankem a překážkou. Pokud je vzdálenost větší, než požadovaný krok tanku, dojde k přemístění na novou pozici. Jestliže je zjištěná vzdálenost menší, je velikost kroku upravena, aby hráč nevstoupil do překážky.

6.5 Multiplayer

Pro realizaci hry více hráčů, jsem vytvořil síťové virtuální prostředí, využívající architekturu klient-server s vlákny. Při jeho návrhu jsem vycházel z [1].

Funkcí serveru je vygenerování náhodných čísel potřebných ke tvorbě scény každého připojeného hráče. Vygenerovaná čísla určují počet překážek ve scéně (počet kostek a válců) a hodnotu semínka, které slouží u každého klienta k vygenerování náhodných souřadnic překážek ve scéně. Server naslouchá na síti a čeká na připojení nového klienta. Po připojení nového klienta ukládá jeho uživatelské jméno a údaje o spojení do seznamu a každému novému klientovi je vytvořeno nové vlákno, které bude přijímat zprávy od „svého“ klienta a rozesílat je ostatním hráčům.

Po navázání spojení se serverem je na straně klienta vytvořeno monitorovací vlákno, které se stará o přijímaní zpráv ze serveru. Pokud je spojení navázáno, klient obdrží ze serveru hodnoty potřebné k sestavení scény.

V následující části jsou vyjmenovány jednotlivé typy zpráv a stručný popis reakce vlákna.

`createUser name x z`

V lokálním světě vytvoří distribuovaný sprite jménem `name` a umístí jej na pozici (`x`, 0, `z`).

`needOtherPlayersDetails A P`

Klient na IP-adrese `A` a portu `P` vyžaduje údaje o lokálním spritu.

`detailsFor username x z r`

Vytvoří distribuovaný sprite jménem `username` a umístí na pozici (`x`, 0, `z`), otočený o `r` radiánů od kladné osy Z.

`username <command> s`

Provede pohyb distribuovaného spritu jménem `username` o krok `s` podle aktuální hodnoty řetězce `<command>`, která může být `forward` (pohyb vpřed), `backward` (pohyb dozadu), `right` (rotace vpravo), `left` (rotace vlevo) a `camRotate` (rotace věže).

`username fire d`

Distribuovaný sprite jménem `username` provede výstřel. Délka dráhy letu střely je `d`.

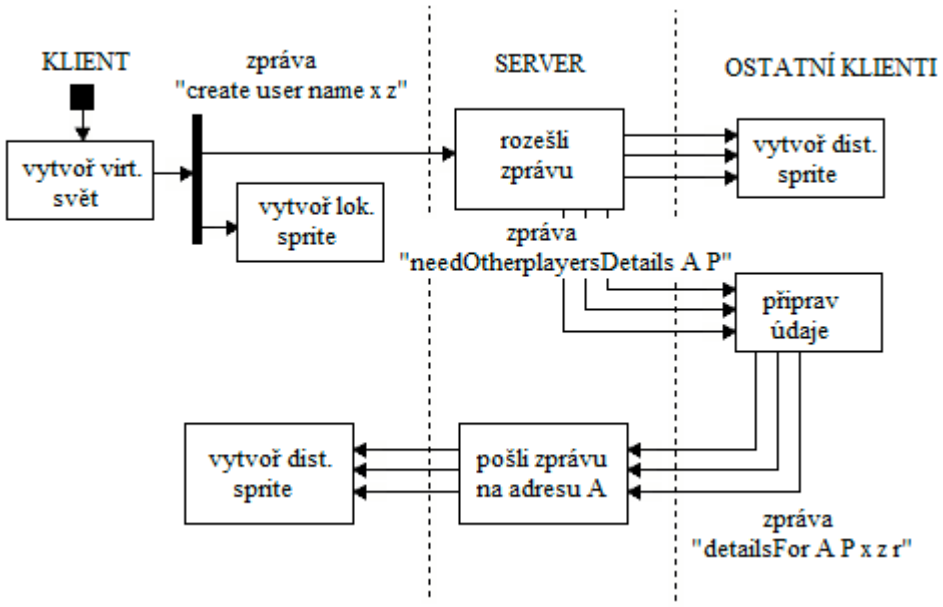
`Username bye`

Klient jménem `username` se odpojil. Vlákno odpojí příslušný distribuovaný sprite z grafu scény.

6.5.1 Tvorba distribuovaného spritu

Pro umístění spritu vybíráme souřadnice z předem vygenerované množiny souřadnic, vygenerovaných náhodně při tvorbě scény. Součástí tvorby lokálního spritu je tvorby zprávy `createUser`, kterou zašleme serveru. Server musí zprávu rozeslat všem aktuálně připojeným klientům. Nový klient musí svou kopii virtuálního světa zaplnit distribuovanými sprity, které představují ostatní uživatele. Tuto činnost zahajuje server odesláním zprávy `needOtherPlayersDetails` všem ostatním klientům. Rozesláním této zprávy se spustí série odpovědí `detailFor`, předávaných serverem zpět novému klientovi. Na základě každé zprávy `detailFor` přidá klient do svého virtuálního světa jeden distribuovaný sprite.

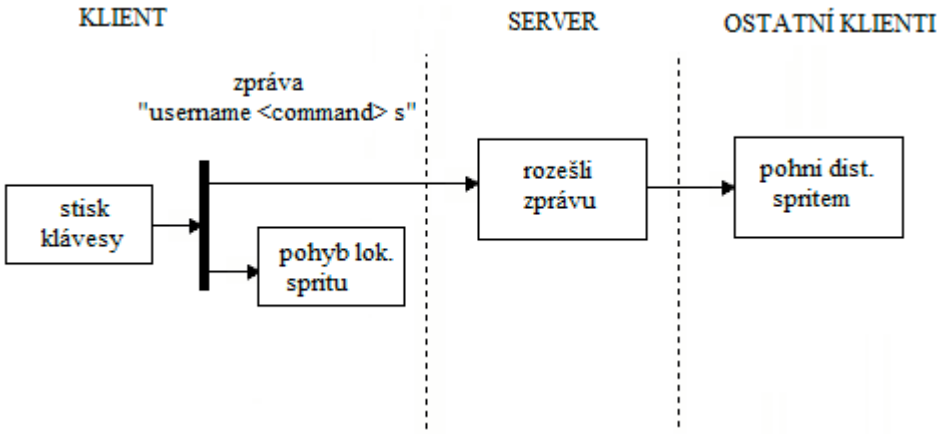
Pro snažší identifikaci hráče je těsně nad ním umístěn 3D text se jménem hráče orientovaný směrem ke kameře.



Obrázek 13. Tvorba lokálního spritu

6.5.2 Pohyb distribuovaného spritu

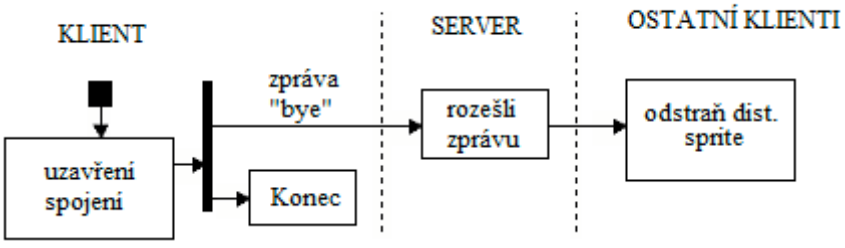
Každá operace pohybu a rotace lokálního spritu je ještě před aktualizací lokální scény doprovázena odesláním zprávy `username <command> s` serveru, který uvědomí ostatní klienty.



Obrázek 14. Pohyb a rotace distribuovaného spritu

6.5.3 Ukončení spojení

Při ukončení hry dojde k odeslání zprávy `bye` na server. Zpráva `bye` způsobí, že server na odchod daného uživatele upozorní ostatní klienty rozesláním zprávy `username bye` a odstraní hráče ze seznamu. Po obdržení zprávy klient odstraní uzel konkrétního distribuovaného spritu z grafu scény.



Obrázek 15. Odchod hráče ze hry

7 Implementace

Aplikace je vytvořená v programovacím jazyku Java ve vývojovém prostředí NetBeans 5.5. Pro práci s 3D grafikou a grafem scény jsem využil knihovny postavené nad jazykem Java, konkrétně jde o knihovnu Java3D ve verzi 1.5.0.

Java3D je aplikační rozhraní pro práci s grafem scény. Nabízí soubor konceptů na vysoké úrovni sloužících k tvorbě, vykreslení a manipulaci s grafem 3D-scény, složené z geometrických útvarů, materiálů, světel, zvuků a dalších prvků.

Při tvorbě scény jsem se pro jednoduchost snažil využívat primitivních objektů implementované přímo v Java3D, jako válce a kostek, které jsem pro lepší vzhled potáhl texturami. Pro reprezentaci tanku jsem se rozhodoval mezi použitím tanku vytvořeného pomocí Java3D nebo externího modelu. Jako externí model se mi nabízel jednoduchý tank vytvořený pomocí programu blender. Pro načtení modelu do scény jsem využil knihovny NCSA Portfolio, která shromažďuje několik rutin pro načítání různých souborových formátů s 3D-daty do jediného, praktického balíčku. Nicméně při jeho zapracování do scény jsem usoudil, že lepší bude přeci jenom použití modelu složeného z primitivních objektů implementovaných přímo J3D. Tank je tak tvořen kvádrem, který tvoří hlavní uzel. Ke kvádru jsou přidány dva válce zaoblující jeho přední a zadní stranu tak, aby tank alespoň částečně připomínal tvarem skutečný tank. Jako vež slouží válec položený na „podvozku“ a dělo je reprezentováno tenkým válcem, připojeného k věži.

K ozvučení hry jsem použil volně dostupné zvukové soubory ze hry Silent Hunter III. Při implementování zvuků do hry jsem narazil na problém s 3D-zvukem, který není v novějších verzích knihovny Java3D podporován. Knihovna musí být volána explicitně při spouštění hry.

Obrázek 16. Zobrazení výsledné hry

8 Závěr

Je všeobecně známo, že dominantními jazyky pro tvorbu počítačových her jsou C a C++. I přesto jsem se rozhodl použít jazyk Java, který v posledních letech zaznamenal v oblasti počítačové grafiky a her obrovský skok dopředu.

Za pomoci odborné literatury jsem si značně rozšířil své znalosti v oblasti programování 3D-grafiky pomocí jazyku Java a naučil se pracovat s knihovnou Java3D, která se ukázala jako dostatečně kvalitní nástroj při tvorbě her.

Práce byla pro mě velikým přínosem, uplatnil jsem při ní získané vědomosti z oblasti počítačové grafiky, her a síťové komunikace.

Cílem práce bylo vytvoření jednoduché počítačové hry pro více hráčů. Pomocí textur a jednoduchých 3D objektů implementovaných v knihovně Java3D jsem vytvořil jednoduchou herní scénu a model tanku. Za použití architektury klient-server a základních technik pro síťovou komunikaci jsem vytvořil jednoduché síťové virtuální prostředí, ve kterém hra probíhá.

Navržená hra určitě neodpovídá potřebám dnešních hráčů, ale posloužila jako základ pro pochopení základních technik tvorby počítačových her a realizace hry pro více hráčů. Tato práce mě velmi zaujala a v budoucnu ji hodlám využít pro aplikování pokročilejších herních a grafických prvků, popřípadě jako základ při tvorbě dalších her i jiných programů, souvisejících s touto tématikou.

Věřím, že tato práce může být přínosem pro všechny, kteří by se chtěli zabývat tvorbou počítačových her.

Literatura

[1] Davidson, A. *Programování dokonalých her v Javě* , Computer PRESS, 2006, ISBN 80-7226-944-5

[2] Silicon Graphics, *OpenGL Programming Guide* [online]. [cit. 2008-04-28]. Dostupné na URL: <http://www.glprogramming.com/red/>

[3] Lidický, B. *Několik poznámek k tvorbě počítačových her* [online]. [cit. 2008-04-29]. Dostupné na URL <http://nehe.ceske-hry.cz/cl_sdl_hry.pdf>

[4] Tišnovský, P. *Grafická knihovna OpenGL: texturování* [online]. [cit. 2008-05-10]. Dostupné na URL <http://www.root.cz/clanky/opengl-22-texturovani/>

[5] Matoušek, P. *Programování sítí TCP/IP.* Přednáška [online]. [cit. 2008-05-03]. Dostupné na URL: < https://wis.fit.vutbr.cz/FIT/st/course-filesst.php/course/ISA-IT/lectures/isa2-sockets.pdf >

[6] Matoušek, P. Síťové aplikace a správa sítí: *Přehled TCP/IP, adresování a konfigurace.* Přednáška [online]. [cit. 2008-05-01]. Dostupné na URL: <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ISA-IT/lectures/isa-uvod.pdf>

[7] Brackeen, D., Barker B., Vanhelsuwé L. *Vývoj her v jazyku Java*

[8] Wikipedie, Otevřená encyklopedie. *Texturování* [online]. [cit. 2008-05-08]. Dostupné na URL: <http://cs.wikipedia.org/wiki/Texturov%C3%A1n%C3%AD>

[9] Wikipedie, Otevřená encyklopedie. *DirectX* [online]. [cit. 2008-05-08]. Dostupné na URL <http://cs.wikipedia.org/wiki/DirectX>

[10] Wikipedie, Otevřená encyklopedie. *Počítačová síť* [online]. [cit. 2008-05-01]. Dostupné na URL:<http://cs.wikipedia.org/wiki/Po%C4%8D%C3%ADta%C4%8Dov%C3%A1_s%C3%AD%C5%A5>

[11] Davidson, A. *Killer game programming* [online]. [cit. 2008-05-04]. Dostupné na URL: <https://fivedots.coe.psu.ac.th/~ad/jg/>

[12] Peringer, P. : *Modelování a simulace.* Studijní opora [online]. [cit. 2008-05-02]. Dostupné na URL: <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IMS-IT/texts/opora-ims.pdf>

[13] Kuželka, O. Java a 3D grafika - světla [online]. [cit. 2008-05-11]. Dostupné na URL < http://interval.cz/clanky/java-a-3d-grafika-svetla/>

Seznam příloh

Příloha 1. CD se zdrojovými texty, spustitelnou aplikací, plakátem a manuálem.