VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

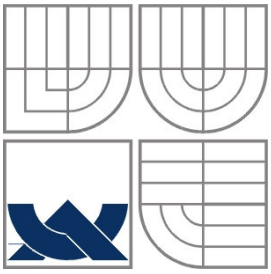
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POČÍTAČOVÁ HRA S VYUŽITÍM ČÁSTICOVÝCH SYSTÉMŮ

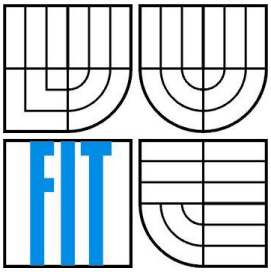
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE LENKA VLKOVÁ
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 S VYUŽITÍM ČÁSTICOVÝCH SYSTÉMŮ

COMPUTER GAME WITH PARTICLE SYSTEMS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

LENKA VLKOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

ING. MICHAL HRADIŠ

BRNO 2008

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací 3D hry s využitím částicových systémů. Částicové systémy se využívají k modelování stochastických přírodních jevů jako například oheň a sníh. V této práci najdete jak popis vývoje této technologie a různá použití, tak i popis konkrétní implementace. Dále jsou v ní diskutována různá témata, která musí řešit tvůrci počítačových her. Mezi tato témata patří výběr herního engine, jež je důležitým a nezbytným nástrojem pro tvorbu hry. Klíčová je detekce kolizí, díky které se mohou objekty pohybovat v prostředí a reagovat na ně. Součástí práce je také návrh implementované hry a popis některých částí její implementace.

Klíčová slova

Počítačová hra, Částicové systémy, Detekce kolizí, Grafický engine Irrlicht.

Abstract

This bachelory thesis describes the design and the implementation of a 3D computer game with particle systems. The particle systems can be used for the simulation of a stochastic natural phenomena, for example fire and snow. In this thesis, you can find the description of the evolution of these systems, their uses and the description of the particular implementation. Other parts of the text are focused on another subjects involved in the game programming. The selection of an essential tool for game creation, a game engine, is crucial. Very important is also a collision detection that enables the movement of the objects in a game world and their responses to it. The design and the description of some parts of the implemented game is also a part of this work.

Keywords

Computer game, Particle systems, Collision detection, Graphic engine Irrlicht.

Citace

Vlková Lenka: Počítačová hra s využitím čističových systémů. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Počítačová hra s využitím částicových systémů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením Ing. Michala Hradiše. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Lenka Vlková
5. 5. 2008

Poděkování

Ráda bych poděkovala svému vedoucímu Ing. Michalu Hradišovi za odborné rady a za trpělivost. Bez něj by tato práce nevznikla. Dále bych chtěla poděkovat grafikovi Janu Šnajdrhonsovi, který mi dodal grafiku a pomáhal s testováním hry.

Obsah

Úvod

V dnešní uspěchané době je více než kdy dříve potřeba najít si čas na tolik potřebnou relaxaci a odpočinek. Někteří lidé sportují, někteří chodí do kina na filmy, někteří čtou, ale mnoho ostatních tráví své chvilky u počítačových a konzolových her. Hry už dávno nejsou jen jednoduchými hříčkami, které na chvíli zabaví nezbedné děti, ale mohou být i plnohodnotným audiovizuálním zážitkem, který dokáže obohatit člověka.

Spolu s technologickým vývojem ale stoupají i nároky, které musí hra splňovat, aby mohla být komerčně úspěšná. Vývoj nejúspěšnějších her na nejnovější konzole může trvat i několik let a je k němu potřeba mnoho odborníků z různých odvětví informatiky. Programátoři grafického jádra, herní programátoři, programátoři fyziky, programátoři umělé inteligence, grafici, animátoři, zvukaři, scénáristé – to je jen několik z profesí, které mohou být při vývoji potřeba.

V této bakalářské práci se budu zabývat vývojem hry Robot Game, kterou jsem navrhla a implementovala. Snažila jsem se, aby to byla hra zábavná a v rámci možností vypadala co nejlépe. Grafika je velmi důležitou součástí hry a dokáže ovlivnit hráče o její koupi. Stejně tak dokáže mnoho hráčů odradit, pokud není dostatečně poutavá a přitažlivá.

Dále se zaměřím na částicové efekty, bez kterých si současné hry již nikdo neumí představit. Co by to bylo za hry bez krásného deště, sněhu, výbuchů a dalších přírodních elementů, které se bez použití částicových efektů dají uvěřitelně vyrobit jen těžce.

V prvních kapitolách se budu zabývat stavem na trhu počítačových a konzolových her, vysvětlím pojem herní engine a popíši grafický engine Irrlicht, ve kterém jsem hru programovala. V dalších kapitolách se budu věnovat částicovým systémům. Zmíním jejich historii, různá použití a techniky, které se při užívání a programování částicových efektů používají.

Po částicových systémech se budu soustředit na samotnou hru Robot Game. Zmíním tituly, které mě při její tvorbě inspirovaly a vysvětlím principy a ovládání v této hře. Nesmí chybět ani informace o tom, jakým způsobem jsem použila částicové systémy.

Tímto končí teoretické kapitoly a začíná popis implementace. V této části vysvětlím, jakým způsobem jsem hru a částicové efekty navrhla a implementovala včetně popisu objektového návrhu.

V závěru nastíním možný vývoj do budoucna, co by se na mojí hře dalo vylepšit a změnit a zhodnotím svoji práci.

1 Počítačové hry

V této kapitole se budu zabývat počítačovými hrami obecně. Zhodnotím současnou situaci na trhu her a zmíním důvody, proč jsem se rozhodla implementovat právě počítačovou hru.

1.1 Současný stav

Počítačové hry jsou součástí informačních technologií již od vzniku prvních počítačů. Od svého vzniku působí také jako pohon pro rozvoj informačních technologií. Tvůrci počítačových her vždy chtěli, aby právě jejich hra byla nejkrásnější, nejpropracovanější a prostě nejlepší. Zvyšující se nároky na spuštění takovýchto her pak vedly k vývoji stále lepšího a výkonnějšího hardware.

V současné době jsou sice počítačové hry na ústupu a vedení přebírají konzolové hry, ale i tak věřím, že počítačové hry budou mít stále svoje místo na herním trhu. Obzvláště pak v žánrech, které si někteří hráči bez počítačových periferií jakou je například myš, neumí představit. K těmto žánrům patří zejména strategické a akční hry.

Při vývoji her musí tvůrce nahlédnout pod pokličku mnoha různých vývojářských odvětví a pochopit mnoho různorodých zákonitostí. Nejvíce se ale vývoj dotýká oboru počítačové grafiky, pod nějž tato bakalářská práce spadá. Grafika hry může dokonce v současné době rozhodnout o úspěchu či neúspěchu určitého titulu a díky skvělé grafice si mnoho v jiných ohledech průměrných titulů vysloužilo cestu do svatyně herní slávy.

1.2 Cíle práce

Cílem této práce je návrh a implementace 3D počítačové hry s využitím částicových systémů. Při návrhu hry by mělo být uvažováno možné ovládání pohybem těla.

Původně jsem měla v úmyslu vytvořit hru logickou, která rozvíjí myšlení, a tak může být hráči i přínosem, nejen relaxací. V průběhu vývoje se obzvláště kvůli co největšímu využití částicových systémů žánr mého titulu změnil na akční. Akční hry ale také mohou být hráčům přínosem, protože rozvíjí reflexy a schopnost rychle reagovat na náhle se měnící podněty.

2 Nástroje pro tvorbu

Druhá kapitola této bakalářské práce se bude zabývat nástroji, které jsem používala při tvorbě počítačové hry Robot Game. Nejprve popíši herní engine a po něm nástroj, ve kterém byla vytvořena grafiky do hry. Program byl vytvořen v nástroji Microsoft Visual Studio 2005 od firmy Microsoft.

2.1 Herní Engine

Jako herní engine se označuje jádro počítačové či konzolové hry. Toto jádro by mělo být schopno vykreslovat 2D či 3D grafiku. Vzhledem ke druhu hry by pak také mělo umět například počítat detekce kolizí, přehrávat zvuky, umělou inteligenci a mnoho dalšího. Cílem herního engine je usnadnit vývoj hry zastřešením funkcí na nižších úrovních.

Volba herního engine je pro vývoj současných titulů kritická. Vývoj vlastního engine je velmi nákladný a mnohdy se menším firmám nevyplatí. Proto mnohé firmy přistupují k licencování některého ze známých a propracovaných herních engine. Jedním z nejznámějších a nejpoužívanějších herních engine pro nejnovější hry pro PC a nejmodernější herní konzole Playstation 3 a XBox 360 je Unreal Engine 3 od firmy Epic Games, který kromě samotného herního jádra obsahuje i mnoho aplikací pro usnadnění práce grafiků a ostatních vývojářů [6].

Obrázek 1 : Hra Gears of War, která funguje na Unreal 3 Engine

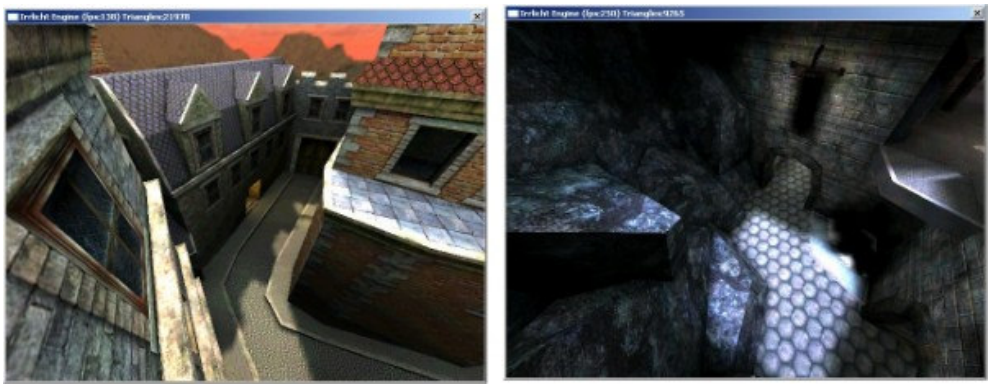
Na poli nekomerčních herních a grafických engine je výběr také bohatý. Mnoho nadšenců pracuje na svém vlastním engine pro různé typy her. Oproti komplexním komerčním engine se ale setkáme spíše s engine zaměřenými na určitou část herního engine. Například můžeme najít pouze grafický engine, zvukový engine či engine na herní fyziku.

2.2 Irrlicht

Pro hru Robot Game byl zvolen 3D grafický engine Irrlicht. Tento engine je stále ve vývoji a s každou novou verzí přináší nová vylepšení a funkce. Během vývoje Robot Game byla uvolněna

verze 1.4, kterou jsem při tvorbě hry použila. Projekt Irrlicht vznikl již v roce 2002 a za jeho počátkem stojí jediný člověk, Nikolaus Gebhardt [5].

Irrlicht sice není herní engine se všemi jeho součástmi, ale obsahuje kromě možnosti vykreslování grafiky pomocí různých renderovacích zařízení, mezi nimiž nechybí OpenGL či Direct3D, i mnoho moderních grafických efektů a jednoduchý systém detekce kolizí. Podporuje také mnoho různých jak texturových, tak i 3D grafických formátů.



Obrázek 2 : Příklad aplikací v Irrlichtu

Mezi jednoznačné klady Irrlichtu patří jeho otevřený kód a to, že je zcela zdarma. Podporuje také různé platformy (Windows 98, Windows XP, Linux, Mac OS a další) s pouze minimálními změnami v kódu. Je relativně rychlý a snadný na naučení. Kolem jeho vývoje a jeho využívání pro různé projekty se vytvořila početná komunita programátorů, která je ochotná zodpovídat dotazy a pomáhat s chybami v kódu.

Irrlicht sice trochu zaostává za nejnovějšími komerčními herními engine, ale na poli nekomerčních bezplatných engine, se mu jen máloco vyrovná (viz Obrázek 2). Obzvláště díky jeho efektivnosti, široké škále implementovaných efektů a snadnému používání, je k programování nejen her velmi vhodný.

Používání Irrlichtu k vykreslování jednoduché 3D scény je velice jednoduché. Nejprve je třeba vytvořit takzvané zařízení (anglicky device), které je kořenovým a nezbytným objektem k používání Irrlichtu. Při jeho vytvoření vznikne i manažer scény, který spravuje veškeré objekty ve scéně, správce uživatelského rozhraní, objekt pro příjem událostí a další objekty potřebné pro chod aplikace.

Objekty, které jsou součástí se označují jako scénické uzly (scene node) a je jich mnoho druhů. Patří mezi ně například 3D model (mesh), kamera, osvětlení, text, ale i například různé efekty jako vodní povrch a další. Irrlicht si na každý objekt, který je do manažeru přidán, uchovává referenci a pokud je třeba, sám se postará o jeho smazání. Samotné vykreslení kompletní scény je pak záležitostí jednoho nebo dvou (v případě použití prvků uživatelského rozhraní) volání. Díky mnoho optimalizacím a využívání stromových struktur v manažeru scény je vykreslování efektivní.

2.3 IrrKlang

IrrKlang je multiplatformní zvukový engine tvůrce Irrlichtu, který se dá snadno propojit s Irrlichtem a který se využívá k přehrávání zvuků a hudby ve hře. Podporuje mnoho různých zvukových formátů včetně populárního formátu mp3 a pro nekomerční využití je zdarma. Proto jsem se rozhodla dodat ke hře i některé zvuky a hudbu, k jejichž přehrávání používám právě IrrKlang.

2.4 Grafika

Pro tvorbu grafiky do hry Robot Game byl použit komerční software 3D Studio Max. Tento software má velkou tradici ve tvorbě modelů, vizualizací i animací do filmů i her. Spolu se software Maya patří 3D Studio Max k nejpoužívanějším software při tvorbě her.

Irrlicht podporuje mnoho formátů pro 3D grafiku. Mezi nimi je i 3ds formát, ve kterém lze exportovat modely z 3D Studia Max. Tento formát ale neuchovává informace o animacích. Formáty 3D Studia Max, které tuto informaci uchovávají, Irrlicht nepodporoval, proto bylo potřeba použít externí exportér dat do formátu x. Formát x uchovává i informace o klíčových snímcích a o animaci a Irrlicht ho podporuje. Vzhledem k tomu, že při použití animovaných modelů byl celkový vzhled hry mnohonásobně lepší, rozhodla jsem se používat u všech modelů formát x.

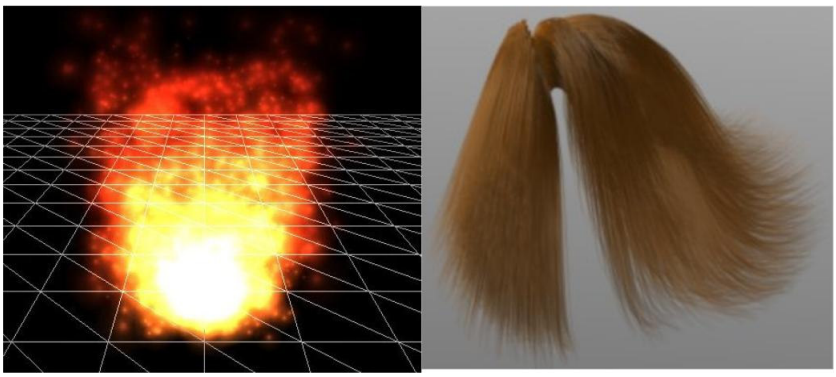
Pro textury částic formát tga především proto, že umí uchovávat i průhlednost (alfa kanál) textury. Stejně jako tga i formát png umí uchovávat průhlednost. Formát png byl použit pro tlačítka uživatelského rozhraní. Pro ostatní textury jsem použila formát jpg, který je výhodný obzvláště díky malé velikosti.

3 Částicové systémy

V této kapitole se seznámíme s pojmem částicové systémy. Nejprve bude vysvětleno, co částicové systémy jsou a k čemu se používají. Bude zmíněn i vývoj této technologie. Na konci kapitoly pak nastíním, jak jsem postupovala při tvorbě částicových systémů do hry Robot Game.

3.1 Úvod do částicových systémů

Částicové systémy jsou metodou simulace náhodných jevů pomocí počítačové grafiky, které nelze realisticky nasimulovat pomocí jiných metod, protože jejich povrch je příliš složitý, členitý či proměnlivý. Těmito jevy mohou být oheň, exploze, kouř, voda, oblaka, ale také například vlasy, tráva, hejna ptáků a další. Povrch a tvar částicového systému není definován polygony. Systém je složen z mnoha 3D bodů v prostoru, které se dynamicky mění a s nimi i tvar systému. Počet částic uvnitř systému se také neustále mění, vznikají nové částice a staré zanikají.



Obrázek 3 : Oheň a vlasy vytvořené pomocí částicových systémů

3.2 Vývoj částicových systémů

Částicové systémy poprvé podrobně popsal William T. Reeves na začátku osmdesátých let dvacátého století. Využil je například k modelování exploze pro film Star Trek II: Khanův hněv. Výbuch planety byl modelován pomocí až 400 částicových systémů, ve kterých bylo celkem až 750 000 částic (viz Obrázek 4) [3].

Obrázek 4 : Výbuch planety ve filmu Star Trek II: Khanův hněv

Základ částicových systémů, který Reeves položil ve své první práci se během let příliš neměnil. Jejich využívání se ale nadále rozšiřovalo. S rozvojem grafického hardware začalo být možné tyto systémy používat i v aplikacích, které fungují v reálném čase. Pokud chceme pomocí částicových systémů dosáhnout realistických výsledků při simulaci přírodních jevů, je potřeba vytvořit obrovské množství částic, které podléhají mnoha složitým fyzikálním zákonům, podle jejichž pravidel se pohybují. Kromě již uvedeného příkladu s výbuchem planety ve filmu Star Trek II například při simulaci realistického kouře za pohybujícím se vozem bylo vytvořeno až 20 000 částic [4].

Pro hry a jim podobné aplikace ale nepotřebujeme dosáhnout zcela realistického efektu, a tak se částicové systémy s úspěchem využívají s nižším počtem částic pro simulaci například deště, sněhu, prachu a mnoha dalších náhodných jevů.

3.3 Základní model částicových systémů

Typický částicový systém se skládá z emitoru a mnoha částic. Z emitoru neustále vznikají nové částice. Emitor může mít různý tvar, který ovlivňuje počáteční pozici jednotlivých částic. Reeves ve svém dokumentu [3] popsal kouli, kruh, rovinu a obdélník, ale tvar emitoru může být prakticky jakýkoliv. Záleží pouze na tom, jakého efektu chceme při vniku částic dosáhnout.

Částicí může být například bod v 3D prostoru, či jednoduchý geometrický tvar, který má mnoho atributů. Mezi typické atributy částic patří:

- pozice
- rychlost
- barva
- energie

Částice ale mohou podle typu systémů a jevu, který systém simuluje obsahovat ještě mnoho dalších atributů. Mezi ně patří například tvar, velikost, průhlednost, předchozí pozice, předchozí barva a další.

Energie částice určuje její životnost v systému. Během existence částice se její ostatní atributy vzhledem k její zbývající energie mohou měnit.

Mezi hlavní výhody modelování objektů pomocí částicových systémů patří fakt, že pomocí několika málo změn v atributech a nastavení, lze dosáhnout zcela odlišných výsledných efektů.

3.4 Životní cyklus částicového systému

Životní cyklus částicového systému lze rozdělit do tří základních fází. Těmito fázemi jsou emitace nových částic, aktualizace atributů částic a vykreslování částic.

Při emitaci v emitoru vznikají nové částice. Vznik nových částic je kontrolovaným stochastickým procesem. Jejich počet se náhodně mění a jeho omezováním, stejně jako omezováním rozsahů jednotlivých atributů můžeme dosáhnout mnoha rozdílných efektů. Každé částici po vytvoření je třeba přiřadit hodnoty jednotlivých atributů. Přiřazování těchto hodnot je také kontrolovaným stochastickým procesem.

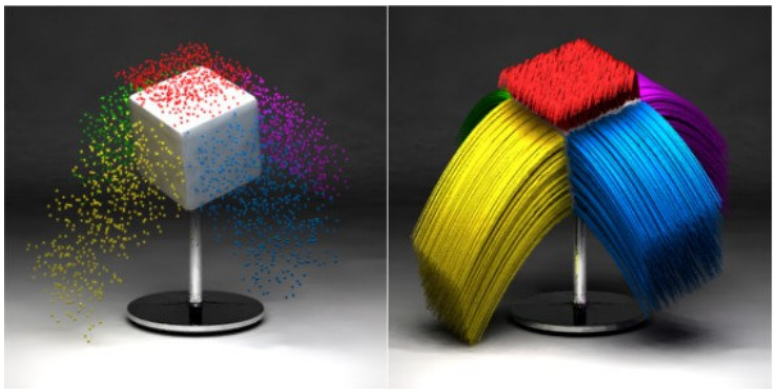
Při aktualizaci atributů částic je třeba nejprve zjistit, zdali některá z částic nedosáhla konce svého života. Tato částice pak musí být odstraněna ze systému. Pro ostatní částice je nutné vypočítat novou pozici. Při výpočtu nové pozice lze využít jak klasických Newtonových fyzikálních zákonů, tak i pouhé přičítání rychlosti k pozici částice. Lze uvažovat například gravitační sílu, která umožňuje docílit parabolické dráhy částic či ovlivňování částic větrem a dalšími reálnými i smyšlenými jevy. V závislosti na druhu systému je potom také nutné změnit další atributy. Některé systémy vyžadují zesvětlování barvy při „stárnutí“ částice, jiné mohou například vyžadovat zrychlování či zpomalování částic.

V poslední fázi se jednotlivé částice vykreslují daným renderovacím zařízením, které máme k dispozici.

Kromě obyčejných částicových systémů, ve kterých je každá částice samostatnou entitou a na ostatní nemá žádný vliv, lze vytvořit i částicové systémy druhého řádu (The Second Order Particle Systems), ve kterých se částice ovlivňují navzájem. Díky vzájemnému ovlivňování částic lze docílit více variabilního a živoucího systému, například ohně. Vzájemné ovlivňování se projeví v životním cyklu systému, ve kterém se po aktualizační fázi objeví ještě fáze vzájemného ovlivňovaní [2].

3.5 Využití částicových systémů

Částicové systémy se dají rozdělit na statické a dynamické. U dynamických se částice pohybuje a vykresluje se pouze nejnovější pozice částice. U statických se ale vykresluje celý její životní cyklus (viz Obrázek 5).



Obrázek 5 : Dynamický a statický částicový systém s krychlovým emitorem

Dynamické částicové systémy se dají využít k zobrazování jevů jako například oheň, voda, déšť a podobně. Je jimi možno simulovat například šíření ohně při požáru. Výsledky takovýchto simulací mohou například sloužit ke zlepšení zabezpečení proti požárům [7].

Statické částicové systémy se využívají k simulaci vlasů, chlupů, rostlin a dalších podobných jevů.

Samozřejmostí je využívání obou druhů v počítačových hrách a ve filmech. Při využívání ve filmech nejsou tvůrci limitovaní výkonem a velikostí paměti grafických karet, a proto dosahují mnohem realističtějších výsledků. V nejnovějších hrách poslední generace pro konzole Playstation 3 a XBox 360 a pro nejnovější počítače se ale již tyto rozdíly pomalu začínají smazávat.

3.6 Animované textury

V počítačových a konzolových hrách se místo barevných bodů využívají i vhodně zvolené textury s průhledností. Při použití textury s průhledností lze dosáhnout mnoha různorodých tvarů částic a přitom se stále vykresluje jen jeden obdélník (případně čtverec), na kterém je textura umístěná. Některé jeho části jsou ale průhledné, a tak ve výsledném obrázku vidíme například kruh.

Při využití textur lze docílit mnohem lepších výsledků při menší výpočetní náročnosti. K dalšímu zlepšení výsledného vzhledu se využívají animované textury. Tyto textury se během života částice mění. Na obrázku (viz Obrázek 6) můžete vidět animovanou texturu pro kouř. Tato textura má osm snímků, které se v průběhu života každé částice přehrají. Černá barva v obrázku ve výsledném vykreslení nebude vidět, bude průhledná.

Obrázek 6 : Animovaná textura

4 Návrh hry – Robot Game

Ve čtvrté kapitole bych chtěla popsat hru, kterou jsem navrhla a implementovala. Popíši také její ovládání, cíle hry a jakým způsobem hra využívá částicové systémy.

4.1 Inspirace

Při návrhu hry jsem se nejvíce inspirovala dvěma tituly. Prvním z nich je stařičká hra SkyRoads, jejíž variaci můžete vidět na následujícím obrázku (viz Obrázek 7 vlevo). Tato hra spočívá v ovládání malé raketky, která umí skákat a jejím cílem je projet až na konec trati. Důraz je ve hře kladen obzvláště na skákání a vyhýbání se různým překážkám. Žánrově by se tato hra dala zařadit jako plošinovka.

Na druhém obrázku (viz Obrázek 7 vpravo) je vidět slavná japonská hra Zone of the Enders od firmy Konami. V této hře je větší důraz kladen na akci. Cíle jednotlivých misí jsou různé, ale v drtivé většině je úkolem něco zničit a nenechat se zničit desítkami nepřátel, kteří neustále útočí.

Obrázek 7 : SkyRoads a Zone of the Enders

Ve hře Robot Game je sice také za cíl dojet až na konec tratě, stejně jako v SkyRoads, nicméně toho lze dosáhnout pouze ničením nepřátel. Podrobně se o cílech hry rozepíši v kapitole 4.3 - Cíl hry. Snažila jsem se dosáhnout herní vyváženosti a také co nejlepší grafiky v rámci možností grafického engine Irrlichtu a výkonu mého počítače.

Při vytváření částicových systémů jsem se inspirovala u nejnovějších herních titulů a filmů. Výjimkou jsou speciální trysky za roboty, které jsou inspirované kresleným seriálem z japonské produkce, Gundam 00 (viz Obrázek 8).

Obrázek 8 : Robot v seriálu Gundam 00 od společnosti Sunrise

4.2 O Robot Game

Žánrově by se hra Robot Game dala zařadit jako akční plošinovka, protože obsahuje prvky obou těchto žánrů. V pozdějších verzích hry ale více převládá spíše akční část hry.

Hra je zasazená v post-apokalyptickém Tokiu a hráč ovládá obrovského robota, který musí projet polorozpadlým městem, aby se zachránil z pomalu umírající planety Země. Cestu mu ale kříží různí mechaničtí nepřátelé, kteří mají za úkol jediné. Tím je mu v cestě za záchranou zabránit.

Cílem hry je dojet v časovém limitu na konec trati. Pokud časový limit vyprší během jízdy, nastává konec hry a hráč musí začít znovu. S časovým limitem, který hráč obdrží na začátku hry je ale nemožné stihnout dojet až nakonec. Proto je třeba sestřelovat různé nepřátele, kteří se na trati objeví. Nepřátel je několik druhů. Za sestřelení největšího nepřítele typu satelit se hráči přičte k časovému limitu deset sekund. Současně je ale třeba dávat pozor na některé nepřátele, kteří také střílejí a na menší nepřátele, které nelze zničit. Po zasažení nepřítelem nastává také konec hry.

Pokud hráč narazí do některé z překážek na trati či do nepřítele, nastává stejně jako v případě zasažení konec hry. Pozor je třeba dávat i na mezery v trati. Ty je možné přeskočit. Pokud hráč sjede z trati a spadne dolů, musí opět začít znovu od začátku.

4.3 Návrh hry

Při návrhu hry jsem vycházela z typických vlastností, která musí každá hra obsahovat. V každé hře nesmí chybět nastavení různých atributů, obzvláště hlasitosti a zvuků, informace o tvůrcích hry a pak samozřejmě samotná hra. Postupně jsem navrhla posloupnost stavů hry, kterou můžete vidět na schématu (viz Schéma 1).

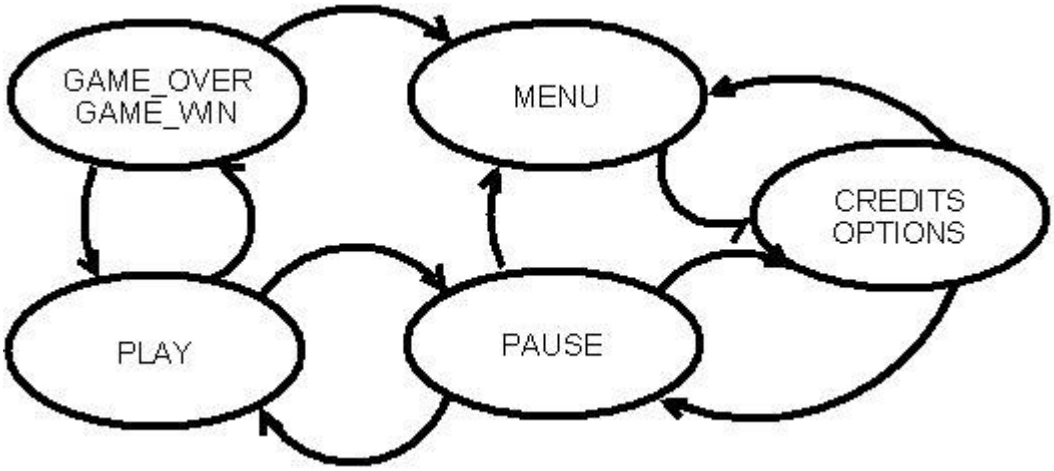


Schéma 1 : Přechody mezi stavy hry

Jednotlivými herními stavy jsou:

- MENU – do tohoto stavu se hra dostane ihned po spuštění
- PAUSE (pozastavení) – stav po pozastavení hry
- PLAY (hra) – stav, ve kterém se hra nachází během hraní
- OPTION (nastavení) – stav pro menu nastavení
- GAME_OVER, GAME_WIN (konec hry, vítězství) – stavy, ve kterých se hra nachází po úspěšném či neúspěšném ukončení hraní
- CREDITS (o autorech) – stav, ve kterém se hráč nachází v menu o autorech

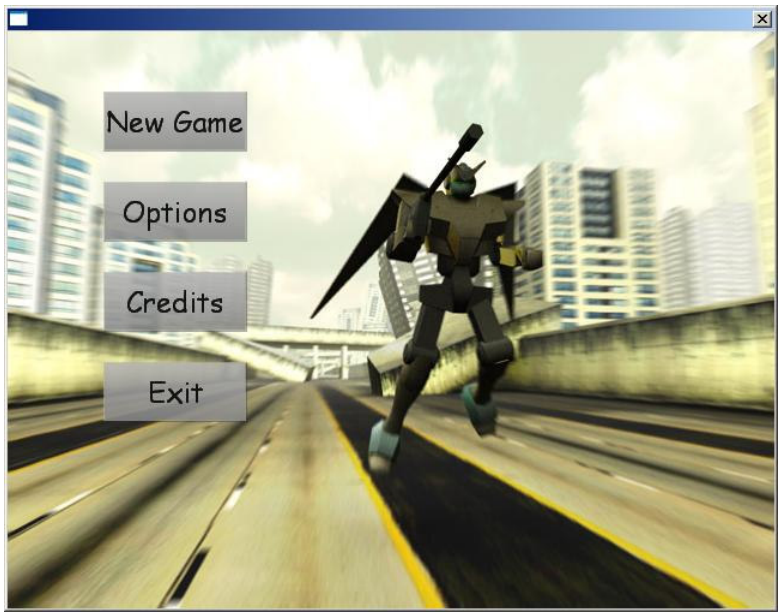
Je důležité, aby se hráč dostal k nastavení jak z hlavního menu, tak z menu při pozastavené hře, což je v tomto návrhu splněno. Z kteréhokoliv stavu je také možné se vrátit do hlavního menu.

4.4 Ovládání

Ovládání hry bylo navrženo tak, aby bylo co nejjednodušší a co nejvíce intuitivní. Inspirovala jsem se ovládáním u akčních her.

Menu se ovládá stiskem levého tlačítka myši. Pokud hráč stiskne v hlavním menu klávesu escape, hra se ukončí. Stisknutím příslušného tlačítka se lze přepnout do nastavení či zahájit hru.

V nastavení (Options) lze nastavovat hlasitost zvuků a hudby a v hlavním menu lze navíc nastavit přepnutí z a na celou obrazovku a zapnout či vypnout částicové systémy u nepřátel. Poslední možnost je vhodná pro slabší počítače. Hra se tím výrazně urychlí.



Obrázek 9 : Hlavní menu Robot Game

Pro pohyb robota, kterého hráč ovládá se využívají šipky. Skok vyvolá hráč stisknutím mezerníku. Pro střílení jsem zvolila klávesu M především díky umístění, které umožňuje její rychlé stisknutí v případě potřeby. Pomocí klávesy C lze přepnout kamera do pohledu z první osoby.

Během hry lze kdykoliv stisknout klávesu escape a hra se pozastaví a přepne do menu.

4.5 Pohyb robota

Pro pohyb robota jsem navrhla jednoduchý fyzikální model. Na první rovnici (1) můžete vidět výpočet odporu prostředí, vektoru **r**. Ten je závislý na vektoru rychlosti. Se zvyšující rychlostí se tak zvyšuje i odpor prostředí ve daném směru. Vektor **r** je také závislý na konstantním vektoru **R**, který zastupuje součinitele odporu a plochu tělesa. V ose y je tato hodnota nastavena na 0,1. V ostatních osách na 0,2. Tyto hodnoty byly zvoleny tak, aby robot po výskoku spadl přiměřenou rychlostí na zem a aby bylo zrychlování realističtější.

$$\mathbf{r} = \mathbf{v} \cdot \mathbf{v} * \mathbf{R} \tag{1}$$

Na rovnici (2) je výpočet gravitační síly $\mathbf{F_g}$, která působí na robota v ose y. V ostatních osách je hodnota nulová. Síla je závislá na hmotnosti tělesa m a na gravitačním zrychlení **g**.

$$\mathbf{F_g} = -m \cdot \mathbf{g} \tag{2}$$

Pohyb robota ovlivňuje také síla pružiny *Fp* (3), která umožňuje realističtější vzhled robotova vznášení. *X* je výchylka pružiny z klidového stavu a *k* je konstanta, která určuje tuhost pružiny. Výchylka se počítá z aktuální pozice robota a tuhost pružiny *k* byla zvolena na hodnotu 1,9. V případě vyšších hodnot bylo pružení robota zanedbatelné a v případě nižších hodnot pružil příliš.

$$F_p = -x * k \tag{4}$$

Výpočet celkového zrychlení **a**, od kterého se odvíjí výsledná rychlost se vypočítá součtem vektorů **mot**, **g** a **r** a přičtením síly F_p v ose y, kde mot je vektor rychlosti motoru robota, **g** je vektor gravitačního zrychlení, r je vektor odporu prostředí a síla F_p je síla pružiny (3). V případě, že hráč stiskl klávesu pro skok, se zrychlení v ose y projeví ve vektoru **mot**, jehož ypsilonová složka se jednorázově zvýší.

$$\mathbf{a} = \mathbf{mot} + \mathbf{g} - \mathbf{r} + F_p \tag{3}$$

Celková rychlost robota (5) **v** v čase t+1 se vypočítá jako součet rychlosti **v** čase t a zrychlení a v čase t.

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{a}_t \tag{5}$$

Konečnou rychlost ovlivňuje ještě odpor pružiny R_p, který je vyjádřen dvěma konstantami. V případě, že působí (robotova pozice je na pružině), jedná se o hodnotu 0,85 a v případě, že se robot nachází nad pružinou, je odpor pružiny 1,0 (6).

$$\mathbf{v}_{t+1} = \mathbf{v}_{t+1} \cdot R_p \tag{6}$$

4.6 Využití částicových systémů

Ve hře Robot Game jsou částicové systémy využívány především ke zlepšení vizuálního dojmu. Pomocí částicových systémů jsou ve hře dělané exploze při zničení robota či nepřátele. Dále se s nimi setkáme u jisker ze střely, kouře z rozpadlých městských trosek a dalších efektů u nepřátel. V neposlední řadě jsou jejich pomocí vyrobeny speciální trysky za roboty (viz Obrázek 10).

Obrázek 10 : Trysky robota ve hře Robot Game

5 Implementace hry

V této kapitole bych ráda popsala implementaci navržené hry. Zmíním se o objektovém návrhu a celkovém průběhu hry. Popíši také, jakým způsobem jsou implementovány některé konkrétní součásti hry.

Celkový průběh programu při spuštění nové hry v hlavním menu je zjednodušeně znázorněn následujícím pseudokódem (viz Algoritmus 1). Jednotlivé fáze programu budou také vysvětleny v následujících kapitolách.

```
1. Inicializace grafického a zvukového engine;
2. Načtení nastavení;
3. Načtení levelu;
4. Herní smyčka {
 Aktualizace hry;
 Vykreslení scény;
 }
```

Algoritmus 1: Průběh programu v pseudokódu

5.1 Objektový návrh

Při návrhů objektů do hry Robot Game jsem se snažila o co nejlogičtější rozdělení jednotlivých prvků hry. Přehled tříd s jejich metodami a atributy najdete na schématu (viz Schéma 2). Nejzřejmější třídou je robot (CRobot), kterého hráč ovládá. Objekt třídy robot má mnoho atributů, mezi nimiž nechybí vlastní pozice, stav, ve kterém se robot nachází, a současná rychlost motoru robota. V atributech nesmí chybět i uzel pro manažer scény Irrlichtu, díky kterému je možné se dostat k dalším potřebným informacím o modelu robota.

Druhou a také klíčovou třídou je třída pro herní level (CLevel). Do objektu této třídy se ukládají především informace načtené ze souboru, který obsahuje informace o levelu. Jedná se o čas, který má hráč k projetí levelu k dispozici, počet nepřátel a pozici, kde se nachází konec levelu. Stejně jako u třídy pro hlavního robota, je i u levelu potřeba uchovávat uzel pro manažera scény, ve kterém jsou uloženy informace o modelu. Pro level je také ukládána informace o jednotlivých trojúhelnících modelu (triangle selector), které slouží k detekci kolizí. V neposlední řadě je potřeba ukládat i informace o nepřátelích, které spravuje také třída pro level. Uzly pro manažera scény jsou ukládány ve vektoru standardní C++ knihovny.

Třetí a poslední třídou mého objektového návrhu je třída pro hru (CGame). Tato třída obsahuje instance objektů obou předchozích tříd a všechny ostatní informace potřebné pro běh hry. Najdeme

v ní proměnné potřebné pro běh grafického (Irrlicht) i zvukového (IrrKlang) engine. Dále obsahuje proměnné, které slouží pro výpočet celkové rychlosti robota a pro výpočet skoku. Mezi další důležité atributy třídy pro hry patří například informace o aktuální kameře. Kamery jsou ve hře dvě. Jedna je z pohledu třetí osoby a druhá, která slouží i ke střílení je z pohledu první osoby. Nepostradatelná je i informace o aktuálním a předchozím herním stavu, stisknutých klávesách či o jednotlivých prvních uživatelského rozhraní, které se v této třídě také nacházejí. Kvůli stabilnímu počtu snímků za sekundu je nutné uchovávat i současný čas, čas vykreslení posledního snímku a informaci o tom, zdali je hra spuštěná v aktuálním okně. Pokud není, je nastavený příznak zastavení hry.

Součástí hlavní herní třídy je i objekt, který přijímá události od uživatele a který na ně vzhledem ke stavu hry a druhu události reaguje.

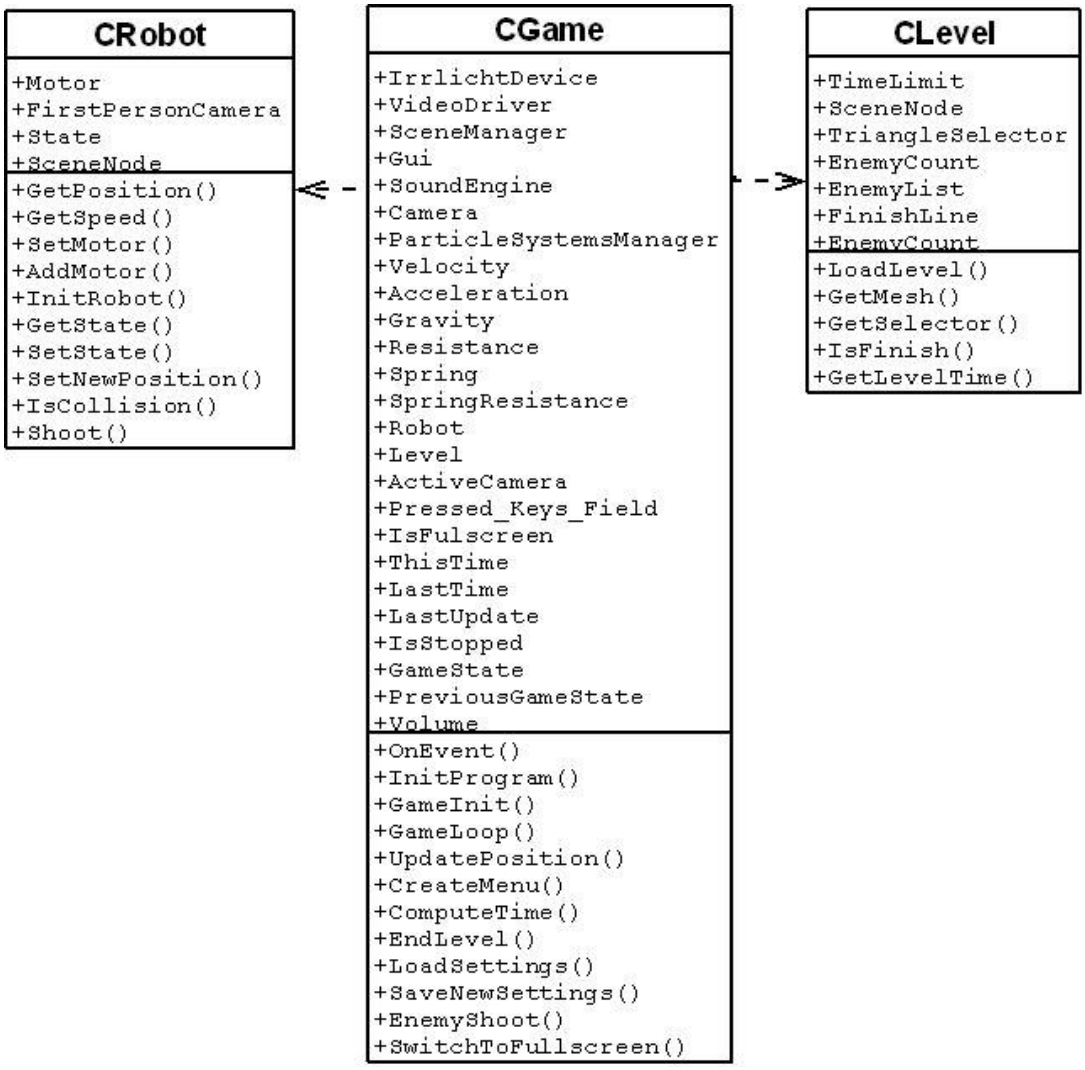


Schéma 2 : Herní třídy se svými metodami a atributy

5.2 Inicializace a načtení nastavení

Ihned po spuštění programu proběhne inicializace grafického a zvukového engine. V případě neúspěchu se program ihned ukončí.

Po inicializaci následuje nahrávání nastavení hry, které se nachází v souboru settings.set. Do souboru se ukládají tři různá nastavení. Prvním je hlasitost zvuků a hudby, kterou reprezentuje desetinné číslo na prvním řádku od 0.0 do 1.0. Druhým je příznak, zdali se má hra spustit na celou obrazovku či nikoliv a je také reprezentován číslem. Tentokrát ovšem pouze celým číslem nula nebo jedna. Třetím nastavením je příznak zapnutí/vypnutí částicových efektů u nepřátel. Toto nastavení je vhodné pro pomalejší počítače a je reprezentováno na třetím řádku souboru nulou či jedničkou.

Nastavení se načítají i ukládají pomocí proudů (streamů). Formát souboru jsem volila tak, aby byl co nejjednodušší.

5.3 Nová hra

Po úspěšné inicializaci a načtení nastavení se hráč nachází v hlavním menu. Z hlavního menu může hráč měnit nastavení (OPTIONS), podívat se na informace o těch, kteří se na vývoji hry podíleli (CREDITS). Dále může hru ukončit, nebo spustit první level.

Po spuštění nové hry je třeba vykonat mnoho operací. Mezi nejdůležitější z nich patří vytvoření instancí tříd level a robot, jejich inicializace, nastavení obou kamer a osvětlení scény.

Načítání důležitých informací o levelu probíhá ze souboru map1.len. Program je s minimální modifikací připraven na načítání libovolného množství levelů z různých souborů a na vytvoření kampaně, ale vzhledem k časové náročnosti vytváření grafiky, je ve hře pouze jeden hratelný level.

V souboru s informacemi o levelu (viz obrázek 11) najdeme nejprve informace o použitých texturách. Na prvním řádku je textura použitá pro okolí levelu (SkyBox). Na dalším najdeme model použitý pro level a na třetím jeho texturu. Následuje informace o časovém limitu v milisekundách. Po časovém limitu přichází na řadu informace o nepřátelích v levelu. Nejprve jejich počet a po něm informace o druhu a pozici každého nepřítele. Na posledním řádku je desetinné číslo, které udává vzdálenost konce levelu od startovní pozice.

```
./textures/GalacticCenter.jpg
./models/road.x
./textures/road.jpg
1200000
5
2
5.1 10.0 10.0
3
20.1 10.0  150.0
1
20.1 10.0  450.0
4
20.1 10.0  650.0
1
20.1 10.0  850.0
10763.0
```

Obrázek 11 : Příklad level souboru

5.4 Jádro hry

Jádrem každé hry je takzvaná herní smyčka (game loop). Herní smyčka se spustí téměř ihned po spuštění programu a dá se říct, že v ní probíhají veškeré operace kromě prvotní inicializace programu. Ukončí se až v případě, že dochází k ukončení programu.

Mnou implementovaná herní smyčka provádí různé operace na základě stavu hry. Hra se může nacházet právě v jednom ze šesti celkových stavů hry. Tyto stavy byly popsány v dřívějších kapitolách.

Ve většině stavů se pouze vykreslují prvky uživatelského rozhraní a čeká se na událost, po jejímž přijetí se stav změní (například kliknutí na tlačítko). Jinak je tomu pouze ve stavu PLAY (hra), ve kterém se provádí aktualizace hry, aktualizace částicových systémů a vykreslení scény. Vzhledem k tomu, že vykreslování celé scény je značně náročné na výkon a že v případě vykreslování „co nejvíce to jde“ by se hra chovala na jinak výkonných počítačích značně odlišně, je vykreslování a aktualizace hry omezeno na určitý počet snímků za sekundu. Scéna se tak vykresluje pouze v případě, že od posledního vykreslení uběhl dostatečný čas.

Vykreslování scény probíhá v závislosti na současném stavu hry. Pokud se hra nachází v některém menu, vykreslují se pouze prvky uživatelského rozhraní. V případě vítězství nebo prohry v levelu se vykresluje celá scéna, stejně jako při hraní hry. Pokud okno, ve kterém je hra spuštěná, není aktivním, vykreslování je pozastaveno. Speciálním případem je stav pozastavení hry (PAUSE), při kterém je zastaven časovač, aby při opětovném spuštění hry nedocházelo k nechtěným efektům (změna pozice nepřátel, posunutí robota a podobně). Pak se pouze čeká na stisknutí klávesy či tlačítka myši.

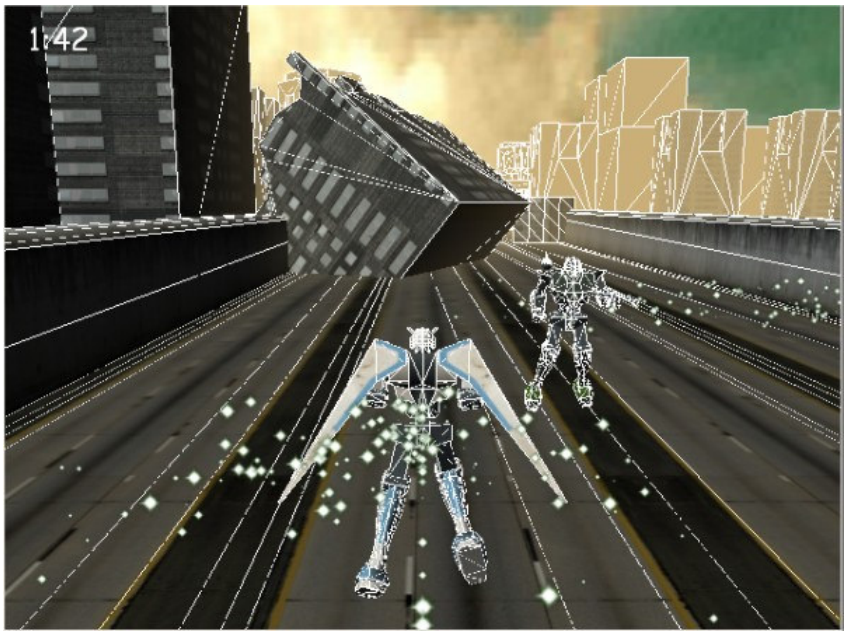
Funkce pro aktualizaci hry (update) se stará o všechny změny scény, které probíhají mezi jednotlivými vykresleními. V této funkci se kontroluje, zda nastal konec hry, počítá se nová pozice robota, nepřátel i kamery. V této funkci se také reaguje na stisk kláves pro pohyb robota (šipek). Na některé z těchto činností se nyní podíváme podrobněji.

5.5 Detekce kolizí

Po výpočtu nové pozice robota je třeba zkontrolovat, zdali mu v cestě nestojí nějaká překážka či nepřítel. S ohledem na druh a pozici překážky je pak třeba novou pozici korigovat. V případě čelního nárazu nastává konec hry.

Irrlicht obsahuje několik vbudovaných funkcí, které jsem k detekci kolizí použila. Ještě před samotným zjišťováním, zda někde nastala kolize, je potřeba připravit objekty, mezi kterými se kolize bude testovat. Irrlicht umožňuje z modelu objektu sestavit takzvaný „výběr trojúhelníků“ (triangle selector). V něm jsou uložené veškeré trojúhelníky, ze kterých se model skládá. Tímto dosáhneme velmi přesného povrchu každého objektu za cenu vyšší početní náročnosti. Pro modely o velkém

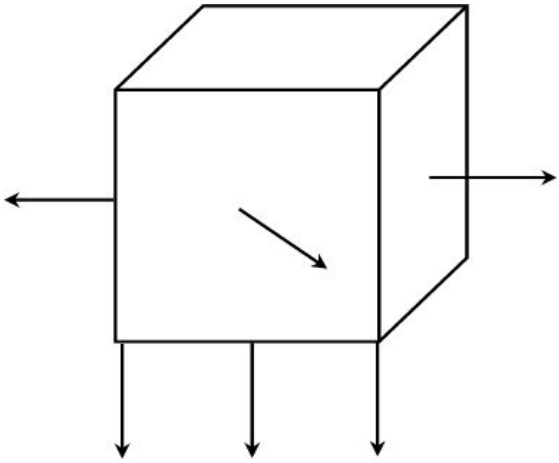
počtu trojúhelníků, jakým je například level, je možné využít triangle selector ve formě oktalového stromu.



Obrázek 12 : Sítě trojúhelníků, ze kterých jsou modely složené

Pro animované modely, jimiž jsou například nepřátele je ale výhodnější využít způsob, kdy se při sestavování triangle selectoru uvažuje i bounding box (hraniční krychle) modelu. Ten se při animaci mění a umožňuje tak vždy sestavit korektní triangle selector.

Samotnou detekci kolizí jsem implementovala pomocí funkce Irrlichtu, která umožňuje najít průnik mezi triangle selectorem a přímkou. Okolo robota jsem umístila několik různě nasměrovaných a dlouhých přímek (viz Obrázek 13), které jsem pak testovala na průnik s trojúhelníky. Tato funkce vrací mimo jiné i trojúhelník, se kterým byl průnik nalezen (pokud byl nalezen). S jeho pomocí se pak dá zjistit naklonění plochy, se kterou robot koliduje. Pokud je plocha méně nakloněná, než je určitý úhel (či spíše určitý rozdíl v jednotlivých souřadnicích vrcholů trojúhelníku), robot nekoliduje, ale jede po ploše nahoru, dolů či šikmo dolů do stran.



Obrázek 13 : Princip detekce kolizí

Nejprve se kontrolují kolize přímek od robota s trojúhelníky levelu. Dolní kolize se kontroluje pomocí přímky, která vede od robota kolmo dolů. Výsledkem kolize je v podstatě zem pod robotem, z jejíchž souřadnic lze vypočítat i například výšku robota nad zemí pro výpočet síly pružiny.

Jako druhá a třetí se kontrolují pravá a levá kolize. Při bočním nárazu nenastává konec hry, a tak je výhodné kontrolovat přední kolizi až nakonec. Pokud nastane pravá či levá kolize, výsledná pozice robota v ose x po aktualizaci zůstává stejná jako při předchozí aktualizaci. Zabrání se tak projíždění robota skrze zdi.

Jako poslední se kontroluje přední kolize. Pokud se před robotem objeví dostatečně kolmá zeď, nastává náraz, výbuch a konec hry.

Po kolizi s levelem je třeba zkontrolovat i kolize se všemi nepřáteli. S nepřáteli není nutné kontrolovat dolní kolizi, ale pravá, levá a přední potřeba je. Při čelní srážce s nepřítelem také nastává konec hry.

5.6 Střelba

Střílení robota probíhá v reakci na událost stisku klávesy M. Nejprve se z atributů kamery z prvního pohledu získá její pozice a cíl, které se použijí jako dráha střely. Následovně se kontrolují kolize s levelem a nepřáteli. Pokud střela před doletem na svou maximální možnou pozici narazí do nějaké překážky, její dolet se zkrátí. Pokud střela narazí do nepřítele, je potřeba se postarat o jeho vymazání a o spuštění částicového efektu výbuchu.

K letu střely jsem využila takzvaný animátor. Tento objekt Irrlichtu umožňuje nastavit uzlu scénického grafu danou trasu, po které se zadaný čas pohybuje, nebo akci, kterou uzel za zadaný čas provede. Irrlicht nabízí například následující animátory:

- Animátor rovného letu (FlyStraightAnimator)
- Animátor pohybu po křivce (FollowSplineAnimator)
- Animátor pohybu po kruhu (FlyCircleAnimator)
- Animátor smazání uzlu (DeleteAnimator)

Pro let střely je využit animátor rovného letu. Ostatní animátory jsou využity například u pohybů nepřátel. Také animátor smazání uzlu po určitém čase jsem hojně používala, například při zničení nepřátel. Animátor pohybu po křivce jsem zase použila na animaci pohybu kamery po úspěšném dojetí na konec levelu.

Abych mohla použít animátor pro let střely a pro smazání případně sestřeleného nepřítele a samotné střely, je třeba vypočítat čas t (7), za který střela k nepříteli doletí. Proměnná l v čitateli zlomku je délka dráhy od současné pozice robota k nepříteli a ve jmenovateli je rychlost v, kterou střela letí.

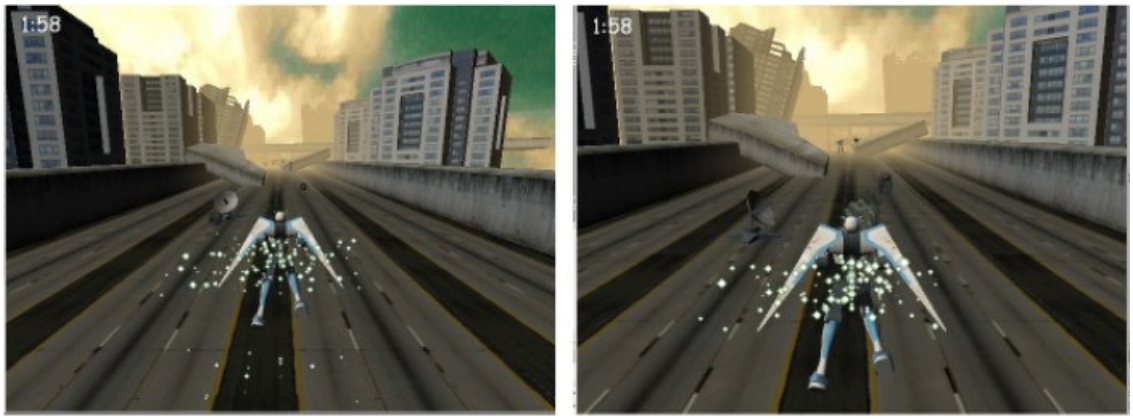
$$t = \frac{l}{v} \tag{7}$$

Střelba nepřátel probíhá při fázi aktualizace hry téměř totožně jako střelba robota. O tom, zda daný nepřítel vystřelí, se rozhoduje na základě náhodného čísla. Pokud je číslo dostatečně velké, nepřítel vystřelí. Funkce, která střílení nepřátel provádí vrací jako návratovou hodnotu, zdali byl robot-hrdina zničen. Hra tak může okamžitě reagovat ukončením hry.

5.7 Zlepšení celkového vzhledu hry

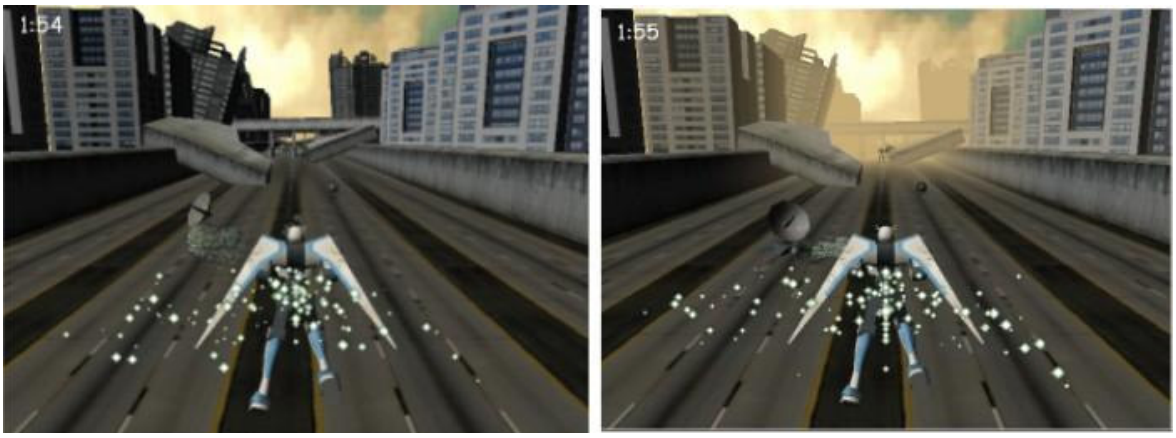
Při programování počítačové hry existuje mnoho možností, jak lze několika nastaveními nebo drobnými změnami v kódu mnohonásobně vylepšit vzhled hry. V této podkapitole bych ráda zmínila ty nejvýznamnější.

Zorné pole (field of view) vyjadřuje velikost prostoru, který lze vidět. Liší se u lidí i zvířat. Jeho nastavení lze hráče více vtáhnout do hry a docílit efektnějšího vzhledu hry. Velikost zorného pole se udává ve stupních. Ve hrách s pohledem z první osoby se běžně používá úhel $90°$. Vzhledem k tomu, že Robot Game využívá pohled ze třetí osoby, zmenšila jsem zorný úhel na přibližně $50°$. Různá nastavení zorného pole kamery můžete vidět na obrázku (viz Obrázek 14).



Obrázek 14 : Zorné pole – vlevo $90°$, vpravo $50°$

V reálném světě nevidíme všechny objekty v dálce ostře. Náš zrak nám to nedovoluje. Ale to počítač netuší, a tak vykresluje i ty nejvzdálenější objekty stejně ostře, jako ty vepředu. To se dá změnit nastavením mlhy (fog). Daleká část scény je v ní zahalená, což vyvolává dojem, že je rozostřená. Hra pak působí více realisticky (viz Obrázek 15). Irrlicht nabízí lineární i exponenciální, vertexovou (počítá se pro jednotlivé vertexy) i pixelovou (počítá se pro každý pixel zvlášť) mlhu. Na obrázku vpravo vidíte výsledné nastavení, které používá lineární vertexovou mlhu. Lineární proto, že vypadá lépe než exponenciální a vertexová proto, že je rychlejší.



Obrázek 15 : Vlevo bez mlhy, vpravo výsledné nastavení

5.8 Změna nastavení hry

V hlavním menu hry najdete položku Options (viz Obrázek 16). Tato položka umožňuje měnit různá nastavení hry. Díky těmto nastavením lze hru přepnout na celou obrazovku, zvyšovat či snižovat hlasitost či vypnout zobrazování částicových efektů u nepřátel. Poslední možnost výrazně zrychlí hru a byla implementována především s ohledem na starší počítače.

Ve stavu OPTIONS, kdy se hra nachází na obrazovce nastavení, lze ovlivňovat posuvník, kterým se nastavuje zvuk či změnit stav zatrhávacího rámečku (checkbox), jehož změnami lze ovlivňovat další dvě nastavení. Všechny tyto změny znamenají vyslání zprávy o změně na prvku uživatelského rozhraní, na které se reaguje ve funkci `OnEvent()`. Tato funkce v Irrlichtu funguje na odchytávání zpráv a je nezbytná pro každý program, který potřebuje na uživatelské podněty reagovat. Pomocí identifikačního označení jednotlivých uživatelských prvků lze lehce rozpoznat, který prvek byl změněn.

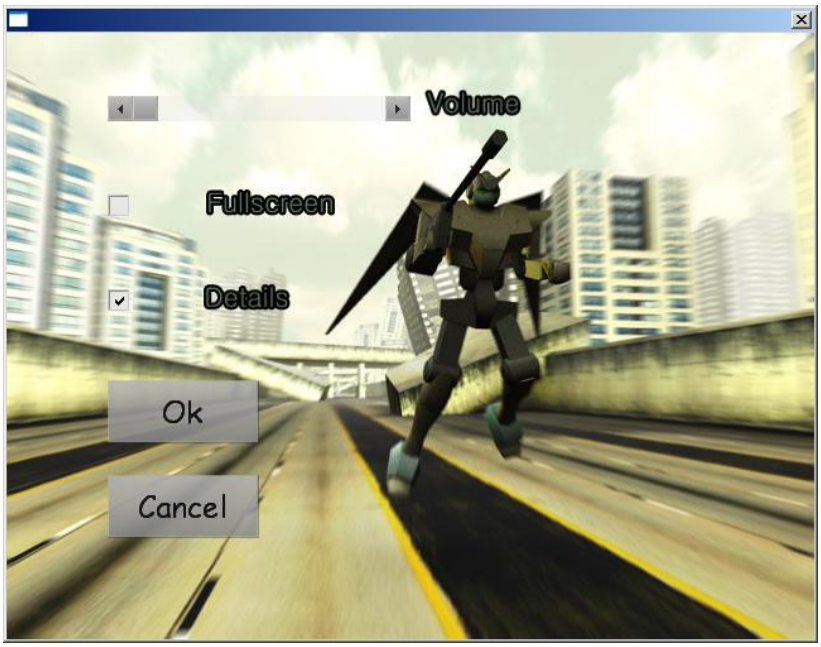
Při změnách na posuvníku je třeba měnit nastavení v programu okamžitě, aby uživatel věděl, jaká bude výsledná hlasitost zvuků a hudby ještě před uložením nastavení a návratem do hry. Konečné uložení nastavení, které se ukládá do externího souboru, se totiž provádí až po stisknutí tlačítka OK.

Další dvě nastavení ale vzhledem ke své povaze nelze provést okamžitě. Vzhledem k tomu, že během spuštěné hry také nelze tyto nastavení změnit, jsou při pozastavení hry tyto nastavení zablokovaná.

Nastavení zapnutí či vypnutí částicových efektů u nepřátel se kontroluje při každém načítání nové hry. Na druhé straně nastavení přepnutí na celou obrazovku se mění pouze při ukládání změn v nastavení. Pokud nastala změna u odpovídajícího prvku (checkboxu), přepne se program do výsledného stavu.

Přepínání z a na celou obrazovku lze v Irrlichtu provést pouze ukončením a zavřením vykreslovacího zařízení (device) a vytvoření nového s požadovaným rozlišením. Po přepnutí je nutné

také znova nastavit klíčové proměnné pro vykreslování scény, jakými jsou například manažer scény či rozhraní pro vykreslovaní prvků uživatelského rozhraní.



Obrázek 16 : Obrazovka nastavení

6 Implementace částicových systémů

V této kapitole vysvětlím implementaci jednotlivých částicových systémů. Také popíši celkový systém vzniku a zániku celých částicových systémů.

6.1 Objektový návrh

Při objektovém návrhu částicových systémů jsem čerpala z materiálů J. von der Burga [1]. V této koncepci jsou základem pro flexibilní a kvalitní částicové systémy tři hlavní třídy. Jedná se o částici, částicový systém a manažer částicových systémů, který vše zastřešuje a přes který se k jednotlivým částicovým systémům přistupuje. Třídy s jejich metodami a atributy najdete ve schématu (viz Schéma 3).

Objekty třídy částice (CParticle) obsahují informace o jednotlivých částicích. Nejdůležitějším atributem této třídy je scénický uzel typu 2D obdélníku, který je vždy natočen ke kameře (anglicky billboard). Scénický uzel v sobě již obsahuje informace typu pozice, velikost a podobně, takže není třeba mít tyto informace uložené i ve třídě částice. Mezi další potřebné atributy patří energie částice a její rychlost.

Ve třídě částicový systém (CParticleSystem) nesmí chybět atribut typu systému. V mé implementaci jsou částicové systémy typu exploze (EXPLOSION), jiskry, kouř, oheň (SPARCLE, SMOKE_UP, FIRE) a trysky (GUNDAM). Mezi další potřebné atributy patří počet aktivních částic, podle kterého se dá zjistit, zdali je celý systém ještě aktivní. Systém se může nacházet ve třech různých stavech. Pokud ještě neuběhl čas do jeho aktivace, který je také jedním z atributů, nachází se ve stavu ještě neaktivní (NOT_ACTIVE_YET). Ve chvíli, kdy začne systém tvořit částice, mění se stav na aktivní (ACTIVE). Jakmile „zemřou“ všechny částice v systému, je systém připraven na smazání a nachází se ve stavu neaktivní (NOT_ACTIVE).

V závislosti na typu systému mají některé typy další vlastní atributy. Pro typ exploze je to příznak, zdali již byl přehrán zvuk exploze a fáze exploze. Podle fáze se určuje počet nově vzniklých částic. V poslední fázi již částice v systému nevznikají a systém pomalu umírá. Pro typ kouř a trysky je třeba uchovávat otcovský scénický uzel, od jehož existence se odvíjí i existence částicového systému, který je na něj navázán.

Ve třídě částicový systém je třeba uchovávat i texturu, která se v tomto systému pro všechny částice využívá a scénický uzel, ze kterého jsou částice emitovány. Jako tento uzel používám neviditelný prázdný uzel (empty node). Samozřejmostí je datová struktura, která uchovává všechny částice systému. Zvolila jsem seznam ze standardní C++ knihovny, který je efektivní při mazání a vkládání. Jeho neefektivita při náhodném přístupu se neprojeví, protože částice se v každém snímku musí procházet všechny v pořadí.

Poslední třídou pro implementaci částicových systémů je manažer částicových systémů (CParticleSystemManager). Manažer má za úkol spravovat všechny částicové systémy, starat se o jejich vznik a zánik. Všechny částicové systémy jsou stejně jako jednotlivé částice uloženy v seznamu standardní knihovny C++. Manažer dále potřebuje uchovávat informaci o scénickém manažeru, který je využíván při přidávání částic do systémů a o zvukovém engine, který přehrává zvuk při explozi. Je nezbytné uchovávat i celkový počet částicových systémů.

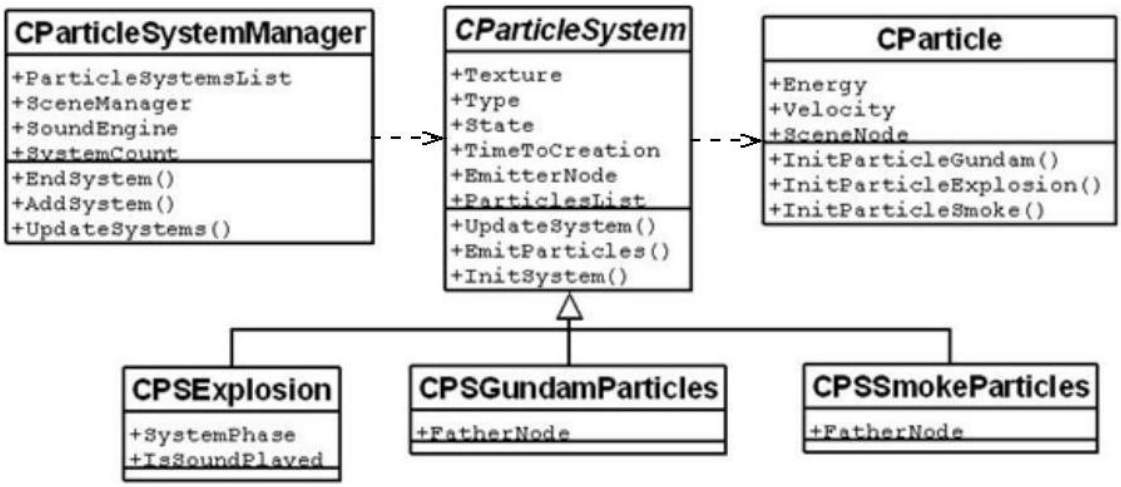


Schéma 3 : Třídy využité v implementaci částicových systémů

6.2 Průběh programu

Vznik, zánik i aktualizaci částicových systémů má na starosti manažer částicových systémů. Ten je volán hlavním programem pokaždé při přidání nového systému a také při každém snímku za účelem aktualizace systémů.

Manažer po požadavku na vznik nového částicového systému vytvoří požadovaný typ a inicializuje ho. Při inicializaci je potřeba vytvořit emitor, nastavit aktivitu a pozici systému a texturu, kterou budou částice v systému používat. Nový částicový systém je po inicializaci zařazen do seznamu systémů. Samotné částice vznikají až při aktualizaci systémů, která probíhá jednou za snímek.

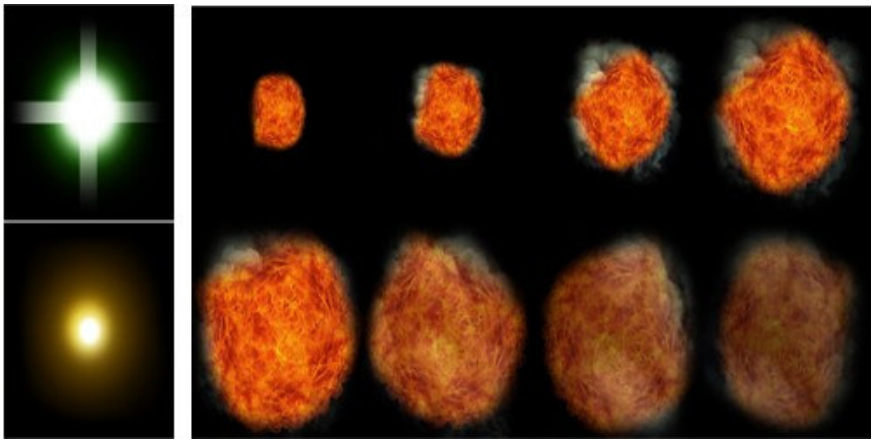
Při aktualizaci stavu hry je volán manažer kvůli aktualizaci jednotlivých částicových systémů. Pokud v některém z nich po aktualizaci nezůstala ani jedna aktivní částice, systém je vymazán ze seznamu.

Během aktualizace konkrétního částicového systému se nejprve zkontroluje, zda již uplynul čas, který byl do počátku emitace nastaven. Pokud ano, emitor začne vytvářet nové částice. Každá částice má podle typu systému náhodné počáteční atributy, které se nastaví při její inicializaci. Pokud se systém nachází v závěrečné fázi (u exploze), nebo již neexistuje otcovský uzel (i trysek, ohně, kouře a jisker), není vytvořena žádná částice. Rozsah počtu vzniklých částic se liší u jednotlivých typů.

Po vytvoření se aktualizují pozice a textury jednotlivých částic. Vzhledem k použití animované textury u exploze, ohně a kouře se musí podle zbývající energie částice u těchto systémů upravit texturová matice, která se nastaví tak, aby byla vidět odpovídající část textury. Pokud klesne energie částice na nulu, je vymazána ze systému. V případě, že jsou takto vymazány všechny částice, přestává být systém aktivní a je také odstraněn.

6.3 Rozdíly jednotlivých systémů

Jednotlivé typy systémů se mezi sebou liší především v rozmezí počátečních nastavení, jakými jsou například velikost, pozice, rychlost a směr rychlosti. Liší se také v použité textuře (viz Obrázek 17). Při aktualizaci systému je také třeba u některých systémů měnit nastavení texturové matice, ale u trysek a jisker toto není třeba.



Obrázek 17 : Textura na trysky robota, jiskry a textura na explozi

6.4 Používání částicových systémů v programu

Používání částicových systémů v programu je jednoduché a intuitivní. Stačí pouze zavolat vytvoření systému, kdykoliv je potřeba a aktualizaci systémů jedenkrát za snímek. Tento částicový systém by bez jakýkoliv úprav mohl být využit pro jakýkoliv jiný projekt v Irrlichtu.

Závěr

Cílem této bakalářské práce byla implementace hry s využitím částicových systémů. I přesto, že by se moje hra Robot Game dala ještě mnoha způsoby vylepšit, jsem s její nejnovější verzí spokojena a myslím, že cíl zadání byl úspěšně splněn. Během práce na této hře jsem se naučila neuvěřitelné množství nových věcí a získala mnoho zkušeností, které věřím, že se mi budou velice hodit v budoucím profesionálním životě. Ráda bych se totiž vývoji her věnovala i v budoucnu.

Během své půlroční práce jsem také moc dobře pochopila, proč hry vyvíjejí celé týmy programátorů. Udělat v současné době kvalitní konkurenceschopný titul totiž není vůbec jednoduché. Věřím ale, že na poli freeware titulů by moje hra po přidání několika dalších levelů konkurenceschopná byla. Veškeré reakce, které jsem při testování hry obdržela, byly kladné.

I přesto existuje mnoho způsobů, kterými by moje hra i její grafická podoba mohla být vylepšena. Jedním z grafických efektů, který by dopomohl k realističtějšímu zobrazení částicových systémů by mohl být takzvaný Motion Blur, který rozmazává pohyblivé objekty tak, aby působily více skutečně.

Dalším z vylepšení by mohla být práce se shadery, které dokáží mnoha druhy efektů vylepšit celkový vzhled hry. V současných hrách a enginech se bez shaderů prakticky nelze obejít. Stejně jako bez různých post-efektů. Ty se aplikují až po vykreslení scény.

Částicové systémy by šly vylepšit přidáním reakcí na prostředí. Částice by mohly kolidovat s prostředím i samy se sebou a případně i osvětlovat svoje okolí. Na poli hratelnosti by se dalo vymyslet mnoho dalších vylepšujících prvků jakými může být například výběr z různých robotů, různé druhy střel, zrychlovací či jiné bonusy, hra po síti a podobně.

Součástí implementace měla být původně i integrace s ovládáním pohybem těla, od které bylo během vývoje upuštěno.

Literatura

[1] VAN DER BURG, J., *Building an Advanced Particle System*. In: Game Developer Magazine, s. 44-50, March 2000.

[2] ILMONEN, T., KONTKANEN, J., *The Second Order Particle System*. In: WSCG Proceedings, 2003.

[3] REEVES, W. T. *Particle System - A Technique for Modeling a Class of Fuzzy Objects*. In: *Proceedings of ACM SIGGRAPH 83*, Computer Graphics Proceedings, Annual Conference Series, s. 359-376, 1983.

[4] CHEN, J. X., WEGMAN, E. J., FU, X., YANG Y. *Near Real-Time Simulation of Particle Systems.* In: 3rd International Workshop on Distributed Interactive Simulation and Real-Time Applications, s. 33, 1999.

[5] *Irrlicht information and documentation*. Dostupné na URL http://irrlicht.sourceforge.net/ (duben 2008).

[6] EPIC GAMES. *Unreal Technology*. Dostupné na URL http://www.unrealtechnology.com/technology.php (duben 2008).

[7] ZHOU, S., SUN, Y., LU, L., CHEN, Z., *Fire Simulation Model Based on Particle System and Its Application in Virtual Reality*. In: Proceedings of the 16th International Conference on Artificial Reality and Telexistence--Workshops (ICAT'06), 2006.

Seznam příloh

Příloha 1

Návod k instalaci a doporučená konfigurace

Pro nainstalování zkopírujte všechny soubory a složky do libovolného adresáře na harddisku. Hru spusťte kliknutím na soubor RobotGame.exe. Ke spuštění není potřeba mít nainstalovaný jakýkoliv další externí program. Hra funguje pouze pod operačním systémem Windows XP.

Minimální požadavky:

Grafická karta s podporou Open GL verze 1.2 a vyšší

Operační paměť: 512MB

Doporučená konfigurace:

Procesor: AMD Athlon 2600+

Grafická karta: Radeon 9600

Operační paměť: 512MB

Program by měl bez větších problémů fungovat i na slabších sestavách. V případě trhání vypněte v nastavení (OPTIONS) zobrazování částicových efektů u nepřátel.

Příloha 2


Počítačová hra - Robot Game
Lenka Vlková
Cíle práce:
Vytvoření počítačové hry s využitím částicových systémů
Výsledky práce:
Robot Game
- několik různých částicových efektů
- detekce kolizí, ničení nepřátel
- skákání, vyhýbání se překážkám
- hudba, zvuky, grafické efekty