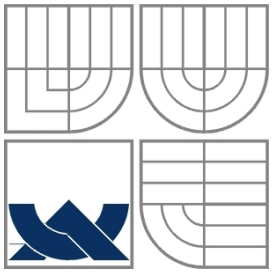
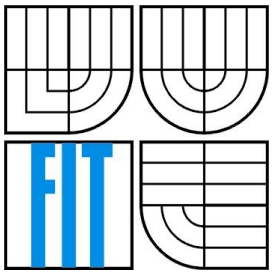


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

GRAFICKÉ INTRO 64KB S POUŽITÍM SLEDOVÁNÍ PAPRSKU

GRAPHICS INTRO 64KB USING RAY TRACING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN KRUPA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ADAM HEROUT, Ph.D.

BRNO 2009

Abstrakt

V mé bakalářské práci jsem mňel za úkol seznámit se s fenoménem grafického intra s omezenou velikostí, prostudovat existující zobrazovací techniky, které využívají sledování paprsku v reálnem čase a za pomocí týchto technik vytvořit vlastní grafické intro, kterého spustitelná verze nebude přesahovat 64kB. Taky popisuji historii těchto grafických inter a důvody ich vzniku. Práce obsahuje popis technik použitých při vytváření intra a taky porovnání těchto technik. Dále popisuje vlastní implementaci aplikace a dosažené výsledky. V závěru jsou pak zhodnoceny přínosy práce a způsoby pokračování práce.

Klíčová slova

intro, demo, demoscéna, sledování paprsku, 3D grafika, rendrováni v reálne čase, odraz, lom svetla, vektor, 64kB

Abstract

The object of my bachelor's thesis was to get acquainted with phenomenon of graphics intro with limmited size, to study actual display techniques, which are using real-time ray tracing and make own graphics intro by using these techniques and executable of this intro is limmited to 64kB. The work also briefly introduces history of making graphics intros and reasons for their creation. It contains description of techniques used to create intro and comparation of them. This work also describes own implementation of aplication and its results. Finally there is evaluation of thesis and ways of improvements.

Keywords

intro, demo, demoscene, ray tracing, 3D graphics, real time rendering, reflection, refraction, vector, 64kB

Citace

Krupa Martin: Grafické intro 64kB s použitím sledování paprsku, bakalářská práce, Brno, FIT VUT v Brně, 2009

Grafické intro 64kB s použitím sledování paprsku

Prohlášení

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

……
Martin Krupa
15.5.2009

Poděkování

Děkuji vedoucímu práce Ing. Adamovi Heroutovi, Ph. D., za cenné rady, informace a podporu při vypracovaní této bakalařské práce.

Obsah

1 Úvod

Tvorba grafických intier patrí do skupiny digitálneho umenia, kde sa zlučujú digitálne a počítačové technológie s umením. Kultúra tvorby intier začala niekedy v osemdesiatych rokoch, odkedy sa datuje vznik počítačového pirátstva. Prvé intrá mali totiž označenie crack intro, alebo tiež cracktro, pretože ich používali crackeri, ktorí odstránili ochranu z nejakej hry, a aby každý vedel kto ochranu danej hry prelomil, vkladali na začiatok svoje krátke grafické prezentácie.

Pod pojmom grafické intro, ktoré sa často nazýva aj demo, si môžeme predstaviť krátke počítačové animované filmy, krátke animácie, alebo aj hudobné klipy. Rozdiel oproti klasickým filmom je hlavne ten, že intro neprehráva uložené dáta ako film, ale po spustení intra sa celá scéna najprv vygeneruje, prepočíta a až potom zobrazí. Jednotlivé scény sú následne prehrávané v reálnom čase, a to všetko pri zachovaní minimálnych rozmerov. Názov demo je odvodený od slova demonštrovať, teda ukazovať a predvádzať schopnosti tvorcov týchto intier, ktorý sa snažia vtesnať svoje schopnosti do čo najmenších veľkostí a tak demonštrovať svoje technické znalosti, umelecký cit a kreativitu. Tvora takýchto intier nie je jednoduchá, pretože sa jedná o animáciu v reálnom čase, a okrem toho je autor obmedzený veľkosťou výslednej aplikácie.

Úlohou tejto práce je predstaviť tvorbu týchto krátkych animácií s obmedzenou veľkosťou a za použitia zobrazovacej techniky sledovania lúčov vytvoriť ukážkovú aplikáciu, ktorej veľkosť nebude presahovať 64kB. Sledovanie lúčov patrí k náročným zobrazovacím technikám, pretože pre zobrazenie jedného snímku scény je potrebné veľké množstvo výpočtov a je veľmi náročná na výkon počítača. Okrem toho nie je podporovaná väčšinou grafických kariet a preto všetky výpočty vykonáva CPU, a nie sú dostupné žiadne knižnice pre túto metódu. Výsledkom je ale veľmi realistické zobrazenie, pretože na podobnom princípe funguje aj ľudské oko. Princípom tejto metódy je vrhanie lúčov zo smeru pohľadu do každého bodu scény a hľadanie priesečníkov s objektmi v scéne. Ak lúč narazí na nejaký objekt, vyšle z priesečníka lúča a objektu ďalšie lúče smerom k svetelným zdrojom a odrazové, lomové a ďalšie lúče, ktoré prispievajú k výslednej farbe bodu. Vďaka tomu je možné vytvoriť obraz ktorý bude obsahovať osvetlenie, tiene, odrazy, zrkadlenie, lom a rozptyl svetla, čím sa veľmi približuje k realistickému zobrazovaniu.

Cieľom mojej bakalárskej práce je tiež vytvoriť grafické intro. Toto intro má obsahovať krátku ukážku animácie za použitia metódy sledovania lúčov. Zobrazovanie má prebiehať v reálnom čase a veľkosť výslednej spustiteľnej aplikácie by nemala prekročiť 64kB.

Na nasledujúcich stránkach najprv podrobnejšie predstavím tvorbu intier, demoscénu, čo je skupina ľudí ktorí sa venujú tvorbe intier, potom rôzne metódy reprezentujúce sledovanie lúčov, popis jednotlivých efektov, návrh a vlastnú implementáciu grafického intra a nakoniec dosiahnuté výsledky.

2 Fenomén grafického intra s obmedzenou veľkosťou

Pod pojmom demo si väčšina ľudí predstaví demoverziu nejakej hry, alebo programu, čiže ukážku, ktorá dáva zákazníkom možnosť vyskúšať si produkt bez nutnosti si ho kupovať. Toto označenie má však ešte jeden význam, ktorý sa od tohto dosť líši. Obidve sú síce skratky od slova demonštrácia v zmysle predviesť niečo, ale zatiaľ čo demá programov do istej miery predvádzajú možnosti komerčných, plných verzií, tie druhé demá, o ktorých hovoríme v tejto práci, sú multimediálne výtvory, ktoré majú iba jediný význam a to predvedenie schopností autorov.

Demo je v podstate program, ktorý neslúži na nič iné iba na prezentáciu tvorby autorov, alebo prezentáciu a demonštráciu techník, pomocou ktorých bolo demo vytvorené. Okrem toho tieto demá väčšinou nemajú ďalšie využitie.

Demo je zvyčajne program, ktorý má veľkosť od 256 bytov do niekoľko megabytov, ktorý po spustení niečo robí. Vo väčšine prípadov je to výstup na obrazovku v podobe obrázkov, či animácií a popri tom hrajúca hudba. Označenie demo alebo intro v podstate znamená to isté. Niekde sa uvádza, že sa jedná o intro, pokiaľ existujú veľkostné limity, a o demo, ak ide o kategóriu bez obmedzenia. Najrozšírenejšie veľkosti intier sú 4kB, 64kB, 128kB a 256kB.

Tieto demá sú väčšinou tvorené nadšencami počítačovej grafiky, ktorý pracujú predovšetkým v skupinách. Jednotlivci sú medzi autormi skôr výnimkou. Každý tvorca sa snaží v rámci stanovených obmedzení vytvoriť čo najlepšie intro. Niektoré skupiny si dokonca vytvárajú vlastné vývojové nástroje a programy na uľahčenie tvorby intier.

Samostatnú kapitolu tvorí hudobná časť demoscény, ktorá je do značnej miery nezávislá. Hudba sa tvorí takzvaným trakovaním v špeciálnych programoch. Vytvorená skladba sa nazýva modul a obsahuje samply (digitálne záznamy jednotlivých nástrojov) a informácie o tom, kedy sa ma ktorý sample hrať. Na rozdiel od klasicky uloženej hudby zaberá oveľa menej miesta, čo je ideálne pre použitie v demách.

Demoscéna existuje na rôznych platformách. Ako prvá vznikla na Commodore C64, a neskôr väčšina initer vznikala na počítač Amiga. Potom sa demoscéna rozšírila aj na PC, a iné osembitové (ZX Spectrum) , či šestnásťbitové (Atari ST) počítače. Dnes sa dajú demá vytvárať prakticky na všetkých platformách, dokonca aj na herných konzolách ako napríklad Dreamcast, Gameboy, Playstation, Xbox, na mobilných telefónoch, a dokonca existujú demá aj na grafických kalkulačkách.

Dávnejšie bol jediným akceptovateľným jazykom pre tvorbu intier čistý assembler, aj keď dnes sa už, hlavne na PC, hojne používajú vyššie programovacie jazyky ako Pascal, C, alebo C++.

Programy používané pri tvorbe intier sú rôzne grafické editory, ako Photoshop alebo 3D Studio MAX. Na vytváranie hudby k intrám sa používajú programy na trackovanie, napr. FastTracker.

2.1 Vznik a história intier

Ako už bolo spomenuté, prvé intrá boli zaznamenané už v osemdesiatych rokoch dvadsiateho storočia. Podnietil ich vznik prvých domácich počítačov, s ktorými vznikali aj prvé programy a hry, za ktoré sa však platilo. Preto súčasne vznikli aj prvý piráti a rozvoj počítačového pirátstva. Hráči ktorí získali hru, prelomili jej ochranu a chceli ju šíriť ďalej, chceli dať ostatným hráčom nejakým spôsobom vedieť, že práve im vďačia za možnosť zahrať si túto hru. Preto sa chceli nejakým spôsobom do hry podpísať. Títo crackeri, jednak aby všetkým ukázali svoje schopnosti, a aby sa vedelo kto prvý prelomil ochranu tejto hry, začali pridávať do hier ako istú formu podpisu svoje intrá. Zobrazovali sa spravidla po spustení hry, a išlo o nejaký zaujímavý grafický efekt, väčšinou jednoduchú animáciu s bežiacim textom, v ktorom autor zdravil, vychvaľoval svoju skupinu, a často urážal tie ostatné. To spôsobovalo súperenie medzi skupinami, ktoré sa predháňali, vtom kto rozšíri viac hier, v rýchlosti ich získania a tiež na kvalite vytvoreného intra, a tak sa niekedy stávalo, že intro vyzeralo lepšie ako samotná hra.

Niektorí sa postupne prestali venovať odblokovaniu hier a začali sa zameriavať práve na tieto krátke programy, ktoré sa do hier vkladali. Títo ľudia sa neskôr od pirátskej scény celkom oddelili, vytvorili si nové skupiny a začali tvoriť iba samostatné intrá, väčšinou na počítačoch C64, neskôr na Amige, ktorej vďačíme za zrod demoscény, tak ako ju poznáme v dnešnej podobe.

Na začiatku deväťdesiatych rokov, keď sa formovala demoscéna aj na PC, tvorcovia intier úplne upustili od nelegálneho šírenia softvéru, a venovali sa už iba tvorbe intier. Dôkazom toho, že demoscéna nemá s pirátstvom už nič spoločné je aj konanie viacerých súťaží, často sponzorovanými samotnými softvérovými spoločnosťami. Tie si uvedomili potenciál tejto zaujímavej zábavy, ktorej sa venuje mnoho schopných a talentovaných ľudí.

Dnes sa už skoro všetky demá vytvárajú na PC, hlavne kvôli podpore hardvérovej akcelerácie a rôznych knižníc, a využívajú 3D grafiku. Rozdiel medzi jedným z prvých intier a súčasnými intrami je vidieť na obrázkoch 2.1 a 2.2.

Obrázok 2.1 Grafické intro Game Music IV vytvorené na Commodore C64 v roku 1985 je jedným z prvých intier

Obrázok 2.2 Scéna z 64kB grafického intra Heaven 7 z roku 2000, 1. miesto na súťaži Mekka & Symposium 2000, ktoré ako jedno z mála využiva ray tracing

2.2 Demoscéna

Demoscéna je spoločenstvo nadšených ľudí z celého sveta, ktorí sa podieľajú na tvorbe grafických intier. Všetci jej priaznivci majú spoločnú jednu vec, a tou je tvorivosť a túžba podeliť sa o svoje výtvory s ostatnými. Členovia demoscény sa rozdeľujú na viacero typov, niektorý programujú, iní tvoria hudbu alebo grafiku. Demoscéna nie je uzavretá komunita, naopak je otvorená každému, kto má záujem a niečo vie. Demoscéneri pochádzajú z radov bežných užívateľov počítačov, ktorí bývajú doslova unesení pri prvom pohľade na niektoré demo, sú ním zaujatí, zháňajú ďalšie a ďalšie, až kým sa nerozhodnú aktívne sa zapojiť. Štandardná demoskupina sa skladá z jednotlivých členov (ktorých počet môže byť od dvoch, až do niekoľkých desiatok členov). Všetci pracujú na základe dobrovoľnosti (je to ich hobby) a každý z nich má presné danú funkciu. Jednotlivé funkcie členov skupiny sú nasledujúce. Najdôležitejšou postavou každej skupiny je programátor, ktorých môže pracovať na jednom deme aj viac. má na starosti samotné programovanie dema, vymýšľa efekty, realizuje ich, vytvára celkový dizajn. Ďalším členom skupiny je grafik, ktorý pracuje a vytvára 2D alebo 3D grafiku použitú v deme. Kreslí bitmapové obrázky, loga, fonty pre texty, textúry pre 3D objekty, navrhuje a modeluje 3D objekty a celé scény, spolupracuje na celkovom dizajne s programátorom. Ďalším členom je hudobník, ktorý vytvára hudobný sprievod k demu. Hudba býva zosynchronizovaná s grafickými efektmi. Posledným členom je swapper, ktorý sa na tvorbe dema priamo nepodieľa, ale zabezpečuje šírenie produkcie do celého sveta.

Jednou z vecí, ktorá značne prispieva k rozširovaniu demoscény je internet, ktorý je hlavným dôvodom rozšírenia demoscény v posledných rokoch. Ten umožňuje zdieľanie vytvorených intier a najdôležitejšou súčasťou sú tzv. demoarchívy, čo sú archívy jednotlivých národných demoscén, alebo demoscény celosvetovej. Každý člen demoscény vystupuje pod svojou prezývkou, a reálne mená často nie sú známe.

Jednou z najväčších a najproduktívnejších demoscén je fínska demoscéna. Najznámejšia demo skupina Future Crew pochádzala práve z Fínska. K ďalším krajinám, kde je tvorba intier rozšírená patria hlavne európske krajiny Švédsko, Nórsko, Dánsko a Nemecko. V Amerike, a iných krajinách sa tvorba intier neteší takému úspechu ako v Európe.

Tvorcovia intier sa každoročne stretávajú na súťažiach, ktoré sa nazývajú demopárty. Na týchto demopárty sa stretávajú scéneri, ako si tvorcovia intier hovoria, kde diskutujú o novinkách vo svete grafických intier a súťažia v rôznych kategóriách. Demopárty prebiehajú vo veľkých miestnostiach, kde má každá skupina počítač a na veľkom plátne sa premietajú jednotlivé demá. O víťazovi rozhodujú samotný účastníci. Vo svete sa koná veľa takýchto podujatí, a najviac z nich v Európe, kde je tvorba dem najviac rozšírená. Medzi najznámejšie súťaže patria dánska The Party, fínska Assembly alebo nemecká Breakpoint. Ceny za víťazstvo v týchto súťažiach sa pohybujú okolo niekoľko tisíc eur, hlavne vďaka bohatým sponzorom a niekedy sa ich účastní aj viac ako tisíc ľudí.

2.2.1 Demoscéna v Čechách a na Slovensku

Tvorba grafických intier nie je v Čechách moc rozšírená. Prvé plnohodnotné české intro vzniklo v roku 1993, ale potom nastal útlm v tvorbe. Až v roku 1998 sa konala prvá demopárty s názvom Fiasko, ktorá sa pravidelne opakovala až do roku 2005. K ďalším známym akciám patrí Marast,, alebo Digital Zooo.

Na Slovensko začalo po demopárty Demobit v roku 1995 obdobie rozvoja slovenskej demoscény. Niektorí tvorcovia dosiahli úspechy aj na medzinárodnej úrovni. Veľmi úspešnou slovenskou akciou, ktorá sa opakuje už 9 rokov je akcia pre osem bitové počítače Forever.

V posledných rokoch dochádza k postupnému rozvoju českej aj slovenskej demoscény a zvyšuje sa aj úroveň vytváraných intier, ktoré bodujú aj v zahraničných súťažiach, predovšetkým v kategóriách s veľkým obmedzením veľkosti, ako 256B alebo 4kB.

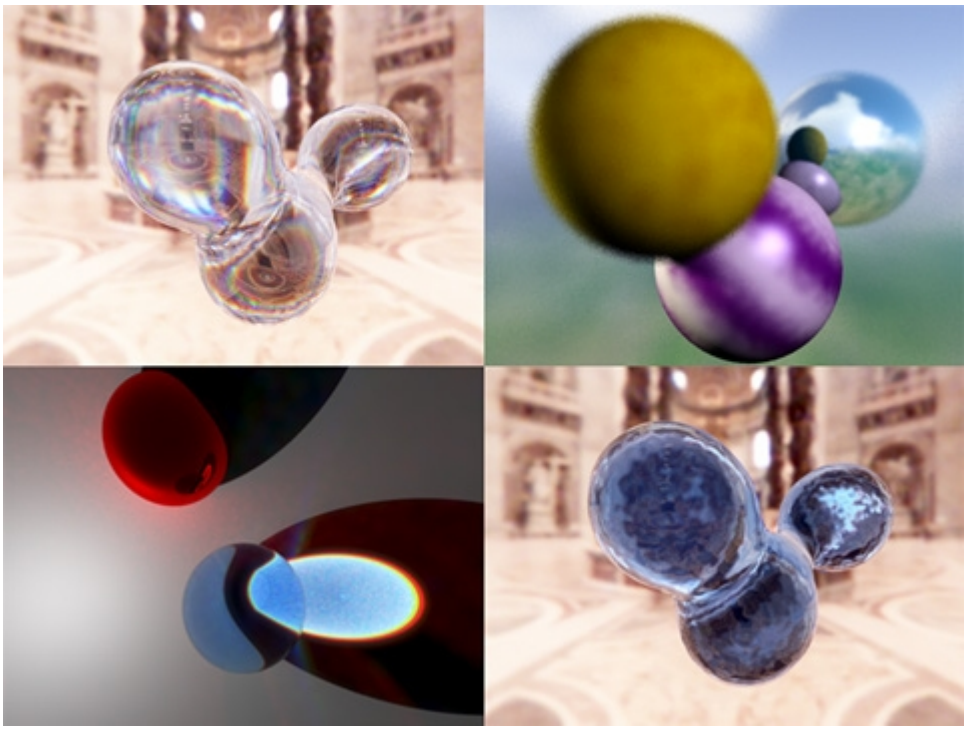
3 Sledovanie lúčov

V tejto kapitole sa budeme venovať zobrazovacej metóde nazývanej sledovanie lúčov, ktorá predstavuje základ celej práce. Hľadanie cesty, ako vytvoriť fotorealistické obrázky na počítači bolo dlho hlavným cieľom počítačovej grafiky. Jedna z prvých úspešných metód na syntézu takýchto obrázkov pochádza z fyziky, presne z jej odvetvia - optiky. Popri vyvíjaní prvých šošoviek, si fyzici často zaznamenávali na papier cestu, ktorú urazil svetelný lúč štartujúci vo svetelnom zdroji a prechádzajúci cez šošovku ďalej. Tento proces sledovania dráhy svetelného lúča bol nazvaný sledovanie lúčov (anglicky ray tracing). Viacerý vývojári počítačovej grafiky, si vtedy mysleli, že táto simulácia svetelnej fyziky je správna cesta na vytváranie počítačom generovaných realistických obrázkov. Mali pravdu, ale nanešťastie v tých rokoch (okolo roku 1960) boli počítače príliš pomalé, aby boli schopné vytvoriť pomocou ray tracingu obrázky, ktoré by vyzerali lepšie ako tie, vytvorené inými, nenáročnejšími metódami. Preto ray tracing upadol do zabudnutia a nebola mu venovaná veľká pozornosť niekoľko rokov. Neskôr s príchodom stále silnejších počítačov, sa pozornosť opäť sústredila na simuláciu reálnej fyziky. Algoritmus ray tracingu bol rozšírený a vylepšený, a umožňoval zobraziť veľa rozličných druhov optických efektov. Dnes je ray tracing jeden z najpopulárnejších a najlepších techník na syntézu obrazu, pretože je jednoduchý a ľahko implementovateľný. Stále ešte existujú aspekty skutočného sveta, ktoré ray tracing nezvláda dokonale (alebo dokonca vôbec) napodobniť, ako napríklad kaustika. Veľa ľudí tvrdí, že ray tracing je jednou z najlepších zobrazovacích techník, ktoré sú v súčasnosti dostupné a s pokračujúcou prácou na jeho vylepšení sa stane ešte viac efektívny a realistický.

Sledovanie lúča označuje skupinu viacerých osvetľovacích a zobrazovacích metód, ktoré sa používajú na prevod reprezentácie 3D scény na 2D zobrazene. Táto technika umožňuje vytvárať veľmi realistické scény, ale za značne vyššieho nároku na výkon. Preto je ray tracing predovšetkým vhodný pre aplikácie, kde môže byť obraz vygenerovaný pomaly, ako sú nemenné obrázky alebo špeciálne efekty vo filmoch. Menej je vhodný pre aplikácie pracujúce v reálnom čase, ako počítačové hry, kde je vyžadovaná vysoká rýchlosť.

V dnešnej dobe má ray tracing široké uplatnenie, najviac v počítačovej grafike, vo filmovom priemysle, ale taktiež našiel využitie aj v architektúre, strojárstve, medicíne a ďalších podobných odvetviach.

Práve kvôli spomínanej náročnosti nie je ray tracing často používaný pri tvorbe grafických intier. Táto technika neumožňuje spracovať príliš rozsiahle scény v reálnom čase, a tak intrá vytvorené prostredníctvom ray tracingu, nie sú ani tak zaujímavé rozsiahlosťou scény a množstvom zobrazovaných objektov, ako svetelnými efektmi spojenými s fyzikálnou optikou (obrázok 3.1).



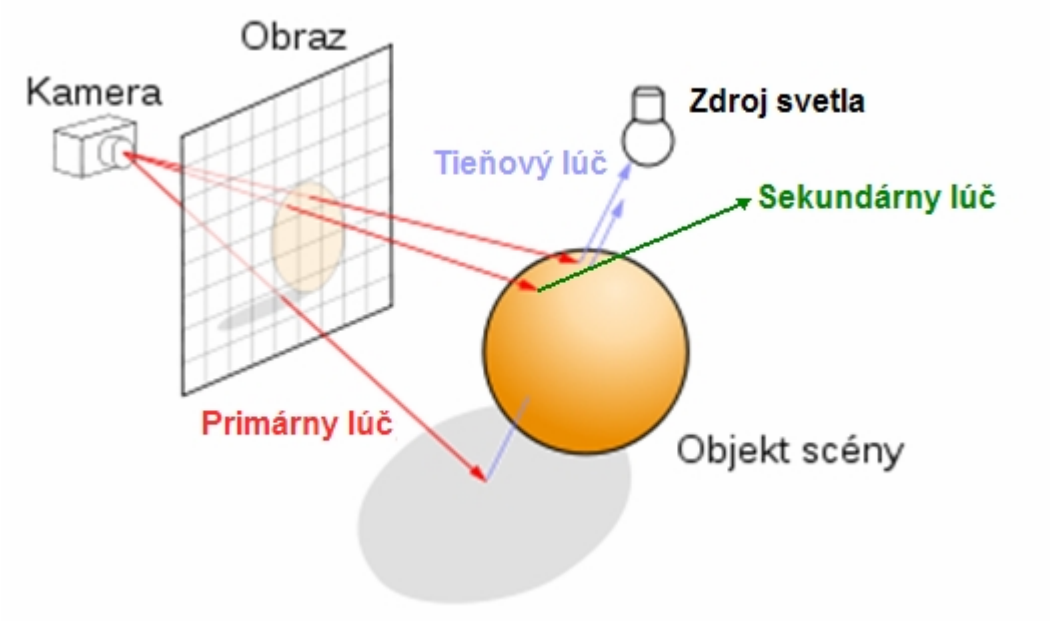
Obrázok 3.1 Rôzne svetelné efekty generované ray tracingom

3.1 Princíp metódy sledovania lúčov

Ako základ pre túto techniku poslúžil reálny svet. V prírode svetelný zdroj vysiela rovnomerne do všetkých smerov lúče svetla, ktoré putujú v priestore, a nakoniec narazia na nejaký objekt, čo naruší ich šírenie a lúče sa buď odrazia, lámu alebo zanikajú. Dalo by sa povedať, že jeden takýto lúč je prúd fotónov, smerujúcich rovnakým smerom. V skutočnosti môže šíriaci sa svetelný lúč zaniknúť, čiže byť pohltený nejakým objektom, alebo sa môže zmeniť jeho smer a to buď ak nastane odraz alebo lem svetla. Povrch objektu môže odraziť všetky fotóny svetelného lúča, kedy dochádza k dokonalému odrazu v jednom alebo viacerých smeroch. Povrch objektu môže tiež pohltiť časť svetelného lúča, čo spôsobuje stratu intenzity odrazeného lúča. V prípade, že je objekt priehľadný, tak dochádza k zalomeniu časti svetelného lúča dovnútra do objektu, absorbovaniu časti intenzity lúča a ovplyvneniu farby lúča. Niektoré z týchto lúčov môžu potom týmto spôsobom doraziť až do nášho oka, kde sú zachytené sietnicou, kde každý bod na sietnici zachytáva farby týchto svetelných lúčov a vytvára výsledný obraz ktorý vidíme.

Táto technika sa nazýva sledovanie lúča, alebo priame sledovanie lúča (anglicky ray tracing, alebo forward ray tracing) Keby sme však chceli tento princíp použiť pri vykresľovaní na počítači, bolo by to veľmi náročné na výkon a na normálnych počítačoch nerealizovateľné. Počet vypočítaných lúčov by bol obrovský a iba nepatrná časť z nich by zasiahla pozorovateľove oko. Ostatné lúče by sa počítali úplne zbytočne, pretože by na výsledný obraz nemali nijaký vplyv.

Preto sa používa technika založená na opačnom princípe, nazývaná spätné sledovanie lúča (anglicky backward ray tracing). Ide vlastne o rovnaký princíp, len s tým rozdielom, že lúče svetla nie sú vysielané zo zdrojov svetla, ale z oka pozorovateľa, a preto sa počítajú iba tie lúče ktoré majú vplyv na výsledný obraz. Pred imaginárnym okom pozorovateľom je akási virtuálna obrazovka a cez každý pixel tejto obrazovky je vrhnutý a sledovaný lúč. Následne je vypočítaná výsledná farba objektu viditeľného cez tento pixel. Táto farba je potom priradená príslušnému pixelu, cez ktorý bol lúč vrhnutý a výsledné pole farebných bodov predstavuje 2D obrázok reprezentujúci 3D scénu. Tento princíp je znázornený na obrázku 3.2.



Obrázok 3.2 Princíp spätného sledovania lúčov

3.1.1 Typy lúčov

Pre jednotlivé lúče sa zaviedlo nasledujúce označenie:

Primárny lúč je vyslaný z miesta pozorovateľa cez pixel obrazu do scény.

Sekundárny lúč je vytvorený po dopade primárneho alebo sekundárneho lúča na teleso. Reprezentuje stav, kedy sa predchádzajúci lúč na povrchu telesa odrazil naspäť do scény, alebo keď prenikol do vnútra polopriehľadného telesa. V danom priesečníku môžeme teda počítať až s dvoma sekundárnymi lúčmi, a to odrazeným a lomeným. Počet sekundárnych lúčov môže byť pri väčšej hĺbke rekurzie oveľa väčší ako počet primárnych lúčov, pretože každý sekundárny lúč môže po dopade na ďalšie teleso spôsobiť vznik nových dvojíc sekundárnych lúčov.

Tieňový lúč je vysielaný z bodu, kam dopadol primárny alebo sekundárny lúč k svetelnému zdroju. Jeho úlohou je zistiť, či sa medzi týmto bodom a konkrétnym zdrojom svetla nenachádza nejaká prekážka zatieňujúca teleso. Pokiaľ nie je nájdená žiadna prekážka, je tento svetelný zdroj

zahrnutý do vyhodnotenia osvetľovacieho modelu v danom bode. Oproti primárnym a sekundárnym lúčom sú výpočty spojené s tieňovými lúčni jednoduchšie, pretože nie je potrebné hľadať najbližšie teleso na dráhe lúča. Výpočet sa môže prerušiť po nájdení akéhokoľvek telesa na dráhe tieňového lúča. Z každého miesta dopadu primárneho a sekundárneho lúča na nejaké teleso je vrhnutých toľko tieňových lúčov, koľko je v scéne zdrojov svetla.

3.1.2 Algoritmus sledovania lúčov

Prvý algoritmus použitý na zobrazovanie pomocou sledovania lúčov bol vytvorený už v roku 1968 Arthurom Appelom. V roku 1972 bol zaznamenaný výrazný pokrok, za ktorý vďačíme Turnerovi Whittedovi. Predchádzajúce algoritmy vrhali lúče z oka pozorovateľa do scény, ale neboli ďalej sledované. Whitted v sledovaní lúčov pokračoval. Ak lúč trafil nejaký povrch, mohol generovať tri nové typy lúčov pre odraz, lom a tieň. Táto nová štruktúra sledovania lúčov viedla k zobrazovaniu realistickejších obrázkov a s menšími úpravami sa používa až dodnes.

Rozlišujeme dve základné varianty algoritmu pre sledovanie lúčov. Prvou je vrhnutie lúča, alebo tiež sledovanie lúča prvého stupňa (anglicky ray casting). Podstatou tejto metódy je, že pre každý bod scény je vrhnutý iba jeden lúč. Zobrazuje sa iba bod na povrchu najbližšieho telesa zasiahnutého lúčom. V tomto bode stanovíme farbu pomocou jednoduchého osvetľovacieho modelu, napr. Phongovho (viď kapitolu 3.5). Vzhľad obrázkov vzniknutým sledovaním lúčov prvého stupňa sa moc nelíši od obrázkov vytvorených jednoduchšími zobrazovacími technikami.

Druhým typom algoritmu je sledovanie lúčov vyššieho stupňa. V tomto prípade sledovanie lúčov nekončí pri nájdení priesečníku s najbližším telesom, ale pokračuje sledovaním ďalších lúčov, odvodených podľa odrazivosti a priehľadnosti zasiahnutého telesa. Aj v tomto prípade opäť vidíme povrch najbližšieho telesa, ale na jeho farbe sa podieľajú aj svetelné príspevky zistené ostatnými lúčmi. Metóda sledovania lúčov vyššieho stupňa dokáže zobraziť na povrchu telies zrkadlové obrazy iných telies. Je tiež schopná spracovať tiene.

Algoritmus klasického rekurzívneho ray tracingu zapísaný pomocou jednoduchého pseudokódu:

```
Pre každý pixel v obraze
    Vytvor lúč z oka cez pixel roviny
    Inicializuj NajbližšíPriesečník na NEKONEČNO a NajbližšíObjekt na NULL
    Pre každý objekt v scéne
      Ak lúč pretína tento objekt
        Ak je vzdialenosť priesečníka t menšia ako NajbližšíPriesečník
          Nastav NajbližšíPriesečník  na vzdialenosť priesečníka t
```

```
    Nastav NajbližšíObjekt  na tento objekt
Ak je NajbližšíObjekt  NULL
    Vyplň tento pixel farbou pozadia
Inak
    Vyšli tieňový lúč ku každému zdroju svetla
    Ak je povrch objektu reflexný, vytvor odrazený lúč (rekurzia)
    Ak je povrch objektu priehľadný, vytvor lomový lúč (rekurzia)
    Vyplň tento pixel výslednou farbou
```

Tento algoritmus popisuje funkciu metódy sledovania lúčov. Opakuje sa pre každý pixel v obraze, pre ktorý vytvorí primárny lúč. Tento lúč sa testuje na prienik s každým objektom v scéne. Ak lúč pretína objekt a vzdialenosť priesečníka je menšia ako vzdialenosť ostatných priesečníkov, nastaví sa zasiahnutý objekt ako najbližší. Ak lúč netrafí žiadny objekt v scéne, výsledná farba pixelu bude farba pozadia. Inak ak lúč narazí na nejaký objekt, tak sa ku každému svetelnému zdroju vyšle tieňový lúč. Ak je povrch objektu, na ktorý lúč narazil reflexný, vytvorí sa odrazený lúč, ak je aj priehľadný vytvorí sa aj lomový lúč. Nakoniec sa vypočíta konečná farba pixelu.

3.2 Výhody a nevýhody

3.2.1 Výhody metódy sledovania lúčov

Medzi hlavné výhody tejto metódy to, že simuluje reálne šírenie svetla v priestore, čo spôsobuje, že táto metóda je schopná generovať veľmi realistický a presný obraz. Efekty ako napríklad odraz a lom svetla, ktoré sú ťažko realizovateľné inými zobrazovacími technikami, sú prirodzeným dôsledkom v metóde sledovania lúčov. Ďalšou výhodou je relatívna jednoduchosť tejto metódy, vzhľadom na kvalitu výstupu. Výpočty pre každý lúč sú väčšinou na sebe nezávislé, preto je táto metóda obzvlášť vhodná na paralelizáciu, kedy sa výpočty rozdelia medzi viac zdrojov a sú počítane súčasne.

Vďaka týmto vlastnostiam má metóda sledovania lúčov veľmi perspektívnu budúcnosť a je čoraz viac rozšírená v počítačovej grafike, ale aj iných oblastiach.

3.2.2 Nevýhody metódy sledovania lúčov

Jednou z najväčších nevýhod tejto zobrazovacej techniky je náročnosť na výkon. Je to spôsobené hlavne počtom lúčov ktoré je treba testovať na priesečníky so všetkými objektmi v scéne. Výpočet týchto priesečníkov môže zaberať až 90% celkového času programu. K ďalším nevýhodám patrí, že podporuje iba bodové svetlá a s tým súvisí aj neschopnosť zobraziť mäkké tiene, pomocou klasického sledovania lúčov sa dajú zobraziť iba ostré tiene. Je to spôsobené tým, že jeden bod môže byť

neosvetlený, čiže leží v tieni, ale jeho vedľajší bod už nemusí mať zakrytý svetelný zdroj a je osvetlený. To spôsobuje ostré prechody zo svetla do tieňa a naopak. Odrazové a lesklé plochy síce odrážajú okolie, ale neodrážajú svetlo do okolia, takže nie sú sekundárnymi zdrojmi svetla. Pomerne veľkou nevýhodou je, že pri akejkoľvek zmene v scéne sa musí znovu vypočítať a prekresliť komplet celá scéna. Klasické sledovanie lúčov nie je adaptívne, čo znamená že zobrazenie prebieha vždy s rovnakým vzorkovaním, nezávisle na situáciu v scéne. Niektoré z týchto nevýhod sa dajú úpravou algoritmu čiastočne alebo úplne eliminovať, ale klasické sledovanie lúčov tieto nedostatky nerieši. Ďalšou nevýhodou je aj to že väčšinou nie je podporovaná grafickými kartami. Preto musia byť všetky výpočty vykonávané na procesore počítača a grafická karta zabezpečuje len výsledné vykreslenie na obrazovku.

3.3 Sledovanie lúčov v reálnom čase

Zobrazovanie v reálnom čase v počítačovej grafike znamená predovšetkým plynulé zobrazenie. K tomuto plynulému zobrazeniu je potrebné zobrazovanú scénu prepočítať a vykresliť aspoň 15krát za sekundu.

Ako už bolo spomenuté v predchádzajúcich kapitolách, sledovanie lúčov patrí k výpočetne veľmi náročným metódam a preto je značne pomalé. O zrýchlenie tejto metódy aby pracovala v reálnom čase sa snažia mnohé firmy a vývojári. Prvý krát sa podarilo vytvoriť ray tracer pracujúci v reálnom čase v roku 1986, a jeho autorom je Michael John Muuss. Odvtedy bolo vynakladané značné úsilie na vytvorenie rýchlejšieho ray tracera, pre viaceré účely, ako napríklad interaktívne 3D grafické aplikácie, počítačové hry, filmy ale aj tvorba grafických intier. Dnes už exituje množstvo 3D nástrojov využívajúcich sledovanie lúčov a pracujúcich v reálnom čase. Firma Intel dokonca v roku 2008 predstavila počítačovú hru Enemy Territory: Quake Wars zobrazovanú prostredníctvom sledovania lúčov. Hra bežala v HD (720p) rozlíšení a dosahovala rýchlosť 14-29 obrazov za sekundu, to všetko na 16 jadrovom (4 sokety po 4 jadrách) Tigerton systéme.

Na urýchľovanie sledovania lúčov boli vyvinuté viaceré techniky, ktoré umožňujú plynulé zobrazenie v reálnom čase. Použitie tejto techniky je však stále limitované výkonom počítača.

3.4 Urýchľovacie techniky

Urýchľovacie techniky sa delia do dvoch hlavných skupín. Prvú skupinu tvoria techniky urýchľujúce výpočet priesečníkov lúčov s rôznymi objektmi v scéne, kam patria napríklad vyhodnocovače priesečníkov, obálky, rôzne hierarchické stromy a iné. Druhú skupinu tvoria techniky znižujúce počet vrhaných lúčov, z ktorých najpoužívanejšie sú metóda adaptívneho vyhladzovania a metóda riadenia hĺbky rekurzie.

Osobitnou skupinou je zrýchľovanie prostredníctvom distribúcie jednotlivých výpočtov na viacej častí (procesorov, počítačov), alebo pridanie špeciálneho procesora práve na výpočty spojené so sledovaním lúčov. Tieto techniky sa často kombinujú pre dosiahnutie najlepšieho výsledku.

3.4.1 Techniky urýchľujúce výpočet priesečníkov

K týmto technikám patrí metóda urýchľovania výpočtu pre každý typ objektu zvlášť. Sú to vyhodnocovače priesečníkov, čo sú krátke podprogramy na výpočet priesečníkov lúča s určitou skupinou objektov. Testujú, či je objekt zasiahnutý lúčom alebo nie, čím vylúčia objekt z výpočtu skôr, ako dôjde k skutočnému vyhodnocovaniu priesečníkov. Táto metóda je vhodná pre komplexné objekty, ako parametrické plochy alebo fraktály.

Ďalej do tejto skupiny radíme aj jednoduché triky založené na transformácií riešenia do základnej polohy. Napríklad hľadanie priesečníka lúča s mnohouholníkom počítame tak, že mnohouholník otočíme do roviny rovnobežnej s osou xy. Takto sa dá nájsť rýchlejšie zistiť priesečník týchto objektov pomocou jednoduchej rovinnej geometrie.

Obálky objektov a priestorové hierarchie, patria k základným urýchľovacím metódam. Princíp spočíva v tom, že zložité telesá sú obalené do jednoduchých obálok rozmanitých tvarov, no najčastejšie to býva guľa pretože priesečník s guľou sa počíta najrýchlejšie. Vyhodnotenie testu na priesečník s takouto obálkou býva výrazne rýchlejšie ako vyhodnotenie testu so skutočným telesom. Ak lúč minie obálku, určite minie aj teleso vnútri a netreba sa ním ďalej zaoberať.

Často sa na urýchlenie sledovania lúčov používajú zhora budované hierarchie, predovšetkým kD-stromy, oktalové stromy a BSP stromy. Tieto stromy rozdeľujú, scénu na časti podľa definovaných pravidiel, a vytvárajú strom príslušnosti jednotlivých objektov do týchto častí. Tieto metódy prinášajú urýchlenie výpočtu hlavne pri tieňových a odrazených lúčoch, pretože v týchto prípadoch nie je nutné prehľadávať celý strom od koreňa, ale stačí pokračovať od naposledy spracovávaného uzla.

Vysoký počet tieňových lúčov sa stal podnetom pre zavedenie pamäti prekážok, ktorá sa označuje aj ako svetelný buffer. Každý bodový zdroj svetla je obklopený kockou pokrytou pravidelnou sieťou, ktorá sa nazýva pamäť prekážok. Pri predspracovaní scény sa do každej bunky tejto siete uloží zoznam objektov, ktorých premietnutie na stenu kocky aspoň čiastočne pokryje danú bunku. Ak sa nachádza v zozname objekt, ktorý pokrýva celý povrch bunky, môžeme zo zoznamu odstrániť všetky vzdialenejšie objekty.

3.4.2 Techniky znižujúce počet lúčov

Rovnako ako aj v iných odvetviach počítačovej grafiky, tak aj pri sledovaní lúčov je veľmi výhodné používať koherenciu, čiže informácie získané pri predchádzajúcich výpočtoch.

Koherenciu využíva aj metóda adaptívneho vyhladzovania. Táto metóda predpokladá, že sa v obraze nachádzajú väčšie plochy rovnakej, alebo podobnej farby. Princíp tejto metódy spočíva v tom, že v určitej oblasti sa sledujú lúče iba v niektorých pixeloch a podľa výsledku sa zafarbí priestor medzi nimi priamo bez nutnosti vrhania ďalších lúčov. Väčšinou sa berú do úvahy štvorce bodov, a lúče sú vrhnuté iba pre rohové body, tie sa vyhodnotia a buď sa oblasť medzi nimi zafarbí, alebo dochádza k ďalšiemu deleniu štvorca. Táto metóda síce významne urýchľuje výpočet obrazu, ale vyznačuje sa stratou niektorých informácií a detailov.

Ďalšou metódou znižujúcou počet lúčov je metóda riadenia hĺbky rekurzie. Ak scéna obsahuje veľa matných alebo nereflexných telies, je zbytočné pre všetky počítať príspevky odrazených lúčov. Zisťuje sa príspevok sekundárnych lúčov k intenzite získanej primárnym lúčom. Ak je tento príspevok nižší ako stanovená hranica, generovanie ďalších sekundárnych lúčov sa zastaví. Ich príspevok k farbe primárneho lúča sa považuje za zanedbateľný. Táto technika však nie je vhodná pre scény, ktoré obsahujú vysoký počet reflexných povrchov. V tomto prípade môže výpočet obrazu skôr spomaliť ako zrýchliť.

Posledný prístup je extrémne pamäťovo náročný, ale jeho cieľom nie je iba urýchlenie výpočtu jednotlivého obrázkov, ale zaistenie interaktívnych zmien farebných parametrov scény, textúr a osvetlenia scény. Podstatou tejto metódy je, že výsledný obraz je doplnený úplnými informáciami o tom ktoré objekty sú v danom momente vidieť a ktoré telesá sa odrážajú na ich povrchoch. Pri zmene farby, alebo odrazivosti telies alebo pri zmene intenzity svetelných zdrojov, sa už nevysielajú žiadne lúče, iba sa vyhodnotia osvetľovacie modely pre tie body, ktorých sa zmena týka.

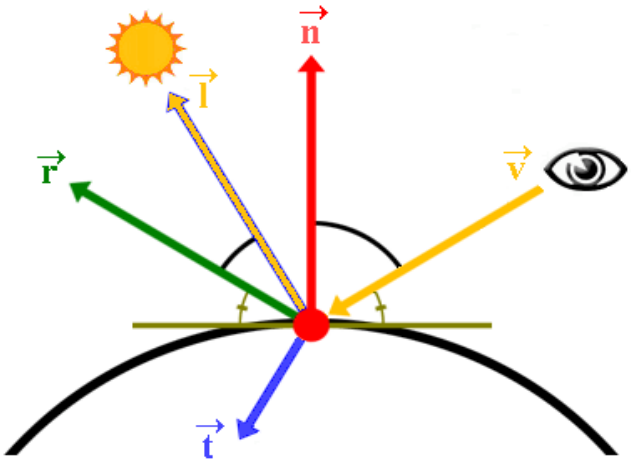
3.5 Osvetlenie scény

Osvetlenie scény definuje osvetľovací model, ktorý určuje ako sa bude v scéne prepočítavať svetlo, ktoré osvetlí objekty a nakoniec dorazí na projekčnú plochu. Existuje veľa druhov osvetľovacích modelov, ale pre metódu sledovania lúčov je najvhodnejší a najpoužívanejší Phongov osvetľovací model. Preto som pre svoju prácu zvolil tento spôsob celkového osvetlenia scény, pretože dosahuje veľmi dobré výsledky pri relatívne nízkej náročnosti.

3.5.1 Phongov osvetľovací model

Ide o empirický osvetľovací model, ktorý navrhol už v roku 1977 Bui-Tuong Phong. Tento model slúži ako náhrada fyzikálneho modelu sveta. Odraz na povrchu osvetleného objektu je určený smerom dopadajúceho svetla $\vec{l}$, smerom k pozorovateľovi, normálovým vektorom v mieste dopadu $\vec{n}$,

a zrkadlovo odrazeným lúčom $\vec{r}$. Phongov model rozlišuje tri druhy odrazu svetla od materiálu a výsledný obraz sa potom skladá z troch zložiek. Sú to zrkadlová, ambientná a difúzna zložka.



$\vec{v}$ - vektor pohľadu
$\vec{r}$ - vektor odrazu
$\vec{l}$ - vektor dopadajúceho svetla
$\vec{n}$ - normálový vektor v mieste dopadu
$\vec{t}$ - vektor lomu

Obrázok 3.3 Phongov osvetľovací model

Zrkadlová zložka predstavuje odlesky svetla od povrchu objektu, ktoré sa nachádzajú na reflexných, hladkých materiáloch. Počíta sa pre každé svetlo oddelene.

Zrkadlová zložka je vyjadrená takto:

$$I = I_L r_s (\vec{v}.\vec{r})^h ,$$

kde I_L reprezentuje farebné zloženie dopadajúceho lúča, $\vec{v}$ je jednotkový vektor pohľadu a vektor $\vec{r}$ vyjadruje smer ideálneho zrkadlového odrazu a je symetrický k vektoru $\vec{L}$ podľa normály. Koeficient zrkadlového odrazu r_s určuje mieru zastúpenia odrazenej zrkadlovej zložky svetla v celkovom odrazenom svetle a jeho hodnota je v rozsahu <0,1>. Koeficient h je skalárny, vyjadruje ostrosť zrkadlového odrazu a udáva sa v rozsahu <1, ∞>.

Ambientná zložka je odrazom bližšie nešpecifikovaného, zo všetkých smerov prichádzajúceho okolitého svetla. Táto zložka nahradzuje z reality to, že každý osvetlený objekt sa sám stáva zdrojom svetla. Okolité rozptýlené svetlo vzniklo mnohonásobnými odrazmi od ostatných telies a rozptylom spôsobeným molekulami vzduchu.

Odraz ambientnej zložky je vyjadrený jednoduchým vzťahom:

$$I_a = I_A r_a ,$$

kde I_A množstvo okolitého svetla, ktoré sa podieľa na osvetlení všetkých povrchových bodov. r_a je farebný koeficient, ktorý vyjadruje schopnosť povrchu odrážať okolité svetlo. Dá sa povedať, že určuje či je teleso svetlé alebo tmavé.

Difúzna zložka určuje farbu vzniknutú priamym pôsobením svetla na objekt. Je počítaná podľa nasledujúceho vzorca :

$$I_d = I_L r_d (\vec{l}.\vec{n}) ,$$

kde r_d je koeficient difúzneho odrazu. Udáva zastúpenie difúznej zložky v celkovom odrazenom svetle a do značnej miery vyjadruje to, čo vnímame ako farbu telesa. I_d je množstvo svetla, a je tým väčšie, čím je smer dopadu bližší normále, čo je vlastne Lamberov zákon difúzneho odrazu $I_d = I.\cos a$. Vzťah má zmysel iba pre $\vec{l}.\vec{n} > 0$, lebo v opačnom prípade je povrch odvrátený od svetla a difúzna zložka svetla I_d je nulová.

Osvetľovací model, používaný pre výpočet farby lúča dopadnutého na povrch telesa, je v algoritme sledovania lúčov upravený a doplnený o ďalšie zložku, ktoré vyjadrujú odrazené a lomené lúče. Sú to zložka I_r pre odraz, a I_t pre lom. Farba I_R lúča prichádzajúceho zo smeru odrazu je ovplyvňovaná koeficientom zrkadlového odrazu r_s telesa. Platí teda vzťah:

$$I_r = r_s I_R$$

Podobne je ovplyvňovaná farba I_t lúča prichádzajúceho zo smeru lomu, ktorá navyše môže byť ovplyvnená ďalším koeficientom, ktorý charakterizuje útlm a závisí na vzdialenosti, ktorú lúč urazil vnútri objektu.

Súčtom všetkých týchto zložiek získame vzťah pre výpočet celkového svetla I_V vnímaného pozorovateľom na povrchu objektu

$$I_V = I_s + I_d + I_a + I_r + I_t$$

4 Návrh a implementácia

Celé intro bolo vyvíjané v operačnom systéme Windows, v programe Microsoft Visual Studio 2008. Samotný program bol implementovaný v programovacom jazyku C++. Assembler alebo jazyk C sú možno pre tvorbu grafických intier vhodnejšie, avšak v dnešnej dobe je použitie C++ pre tvorbu grafických intier stále častejšie. Hlavnou výhodou jazyka C++ je podpora objektového prístupu, ktorý je veľmi vhodný na implementáciu podobných aplikácií.

Pred samotným návrhom som sa musel oboznámiť s tým ako sa vlastne grafické intrá s obmedzenou veľkosťou vytvárajú, pretože ide o môj prvý výtvor v tejto oblasti . Našťastie exituje množstvo kvalitných internetových stránok venujúcim sa tejto problematike. Zoznam týchto stránok, z ktorých som čerpal uvádzam v zozname použitej literatúry.

Mojím cieľom bolo vytvoriť program, ktorý po spustení otvorí okno v ktorom bude zobrazená krátka animácia. Pri tvorbe grafických intier sa nepredpokladá žiadna interakcia s užívateľom, takže odpadlo vytváranie grafického užívateľského prostredia a ošetrenie užívateľských vstupov, až na jediný možný užívateľský vstup, ktorým je umožnenie užívateľovi predčasne ukončiť beh intra.

Celé intro muselo byť vytvorené za použitia techniky sledovania lúčov. Ako už bolo spomínané vyššie, ide o techniku veľmi náročnú na výkon. Preto bolo nutné algoritmus sledovania lúčov optimalizovať a použiť viaceré akceleračné techniky. Samotný ray tracer, čo je program, ktorý uskutočňuje sledovanie lúčov, bol vytváraný postupne. Najprv bol vytvorený veľmi jednoduchý ray tracer, ktorému boli postupne pridávané nové funkcie a efekty, až vznikla výsledná aplikácia.

Pri samotnej tvorbe ray traceru som bol obmedzovaný hneď dvoma limitmi. Prvým bolo, že výsledná aplikácia musela pracovať v reálnom čase. To neumožňovalo vytváranie zložitých efektov, ktoré potrebujú obrovský počet výpočtov a nie je ich možné zobraziť v reálnom čase. Všetky použité techniky museli byť optimalizované, aby pracovali čo najrýchlejšie. Na urýchlenie výpočtu scény som do programu implementoval techniku vyhladzovania, ktorá znižuje počet lúčov vrhaných do scény. Ďalším obmedzením bolo obmedzenie veľkosti na 64kB. To vylučovalo použitie mnohých štandardných knižníc a taktiež musel byť program značne efektívny, aby dokázal čo najviac pri zachovaní minimálnych rozmerov. Aplikácia nemôže používať žiadne dopredu uložené dáta, pretože takto by veľmi ľahko prekročila limit 64kB. Všetky potrebné údaje, ako napríklad objekty alebo textúry musia byť vygenerované až za behu aplikácie. To vedie k použitiu veľmi jednoduchých scén s malým počtom objektov. Veľkosťou je taktiež značne obmedzená dĺžka trvania intra.

Samotný ray tracer je vytvorený podľa návodu z [5]. Časť zdrojového kódu je taktiež prevzatá z tejto stránky. Tento ray tracer je ďalej upravený aby spĺňal vyššie uvedené požiadavky.

4.1 Implementácia

V tejto kapitole je bližšie predstavený princíp hlavných algoritmov programu a princíp ich fungovania. Nie sú tu popísané všetky detaily, ale zameriavam sa iba na tie hlavné, ktoré tvoria podstatu celého programu.

4.1.1 Vektory

Asi najpoužívanejším matematickým prvkom používaným v programe je vektor. Je reprezentovaný triedou *vector3*. Ide o klasický trojzložkový vektor, ktorý obsahuje súradnice x, y, z. Tato vektorová trieda ma naimplementované základné vektorové operácie, ako vektorový a skalárny súčin, normalizácia vektoru, zistenie dĺžky vektoru, súčet, rozdiel a negácia. Vektory sa používajú predovšetkým na reprezentáciu samotných lúčov, kde sa potom počítajú priesečníky tohto vektora s objektmi v scéne. Trieda *vektor3* je taktiež použitá na uloženie farby bodu. V tomto prípade sú namiesto zložiek x, y, z, použité r, g, b, ktoré predstavujú jednotlivé zložky farebnej schémy RGB.

4.1.2 Sledovanie lúčov

Algoritmus sledovania lúčov generuje lúče, ktorými potom testuje na priesečníky s objektmi v scéne. Každý lúč je reprezentovaný bodom v ktorom tento lúč začína a smerom. Základným dátovým typom pre jeden lúč je trieda *Ray*, ktorá obsahuje súradnice počiatočného bod *m_Origin* , a smerový vektor *m_Direction*. Smer lúča je najprv normalizovaný a až potom je samotný lúč zostrojený.

Samotné sledovanie lúča je realizované funkciou *Raytrace*, ktorá najprv volá funkciu *FindNearest*, ktorá testuje na priesečník všetky objekty v scéne a určí, ktorý zo zasiahnutých je najbližší. Farba bodu pre tento lúč sa nastaví podľa farby zasiahnutého objektu. Potom sú vysielané tieňové lúče pre každý svetelný zdroj. Funkcie *CalcShade* a *FindOculder* zisťujú či medzi zasiahnutým bodom a svetelným zdrojom leží nejaká prekážka. Ak je svetelný zdroj viditeľný z daného bodu, je príspevok tohto svetla zarátaný do výslednej farby bodu. Taktiež sa zisťuje, aký je rozdiel medzi normálou zasiahnutého bodu a vektorom smerujúcim k svetlu z tohto bodu, pomocou funkcie *DOT*, ktorá počíta skalárny súčin vektorov. To spôsobuje, že body, ktoré sú priamo osvetlené svetelným zdrojom, sú osvetlené jasne, a odvrátené body tmavo. Ďalej sú rekurzívne sledované sekundárne lúče pre odraz a lom.

Odrazený lúč sa počíta podľa vzorca

$$\vec{R} = \vec{V} - 2(\vec{V}.\vec{N})\vec{N},$$

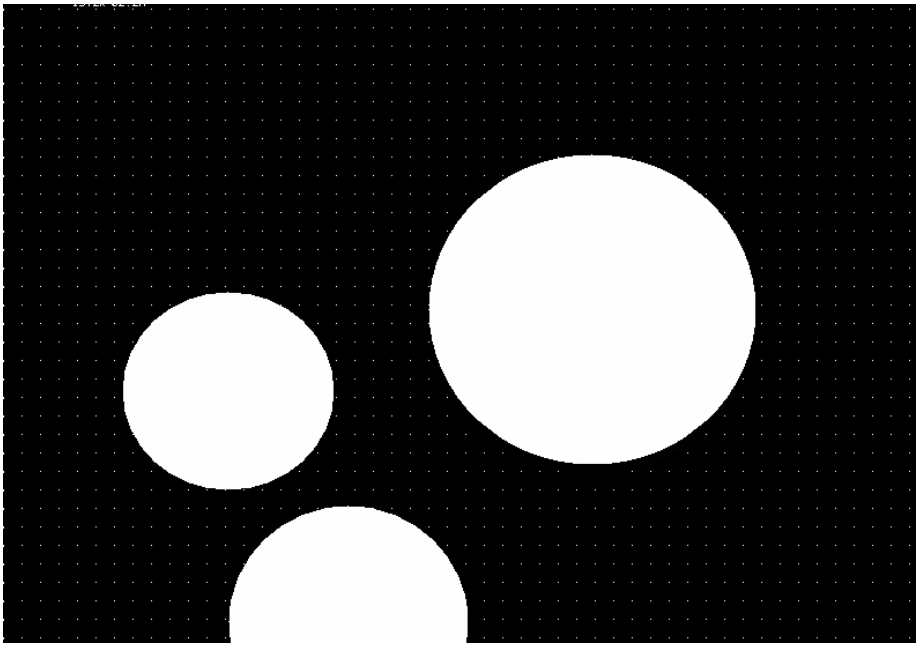
kde $\vec{R}$ je vektor odrazeného lúča, $\vec{V}$ pôvodného lúča a $\vec{N}$ je normálový vektor povrchu objektu.

Lomený lúč sa počíta podľa vzorca

$$\vec{T} = \frac{n_1}{n_2}\vec{E} - \left(\frac{n_1}{n_2}\cos(a_1) + \sqrt{1-\sin^2(a_2)}\right)\vec{N},$$

kde $\vec{E}$ je vektor pôvodného lúča, $\vec{N}$ normálový vektor, a_1 a a_2 sú uhly pôvodného a lomeného lúča a n_1 a n_2 sú indexy lomu pôvodného prostredia a prostredia (objektu) do ktorého lomený lúč vstúpil. Výsledná farba bodu je potom vypočítaná, zohľadňujúc všetky potrebné zložky a príspevky sekundárnych lúčov a uložená.

Ďalej je naimplementovaná zrýchľovacia technika vyhladzovania, ktorá je súčasťou funkcie *RenderTiles*. Výsledný obraz sa počíta po dlaždiciach, čo sú štvorcové polia pixelov o rozmeroch *TILESIZE * TILESIZE*, kde TILESIZE je počet pixelov, podľa toho akú chceme mať dlaždicu veľkú. Najprv sa kontrolujú rohové pixely, cez ktoré sa sleduje lúč a zistí sa ktorý objekt zasiahli. Ak nezasiahli žiadny objekt, tak sa celá dlaždica zafarbí farbou pozadia. Ak však niektorý z týchto štyroch lúčov zasiahne nejaký objekt, sú postupne vrhané lúče celou touto dlaždicou.

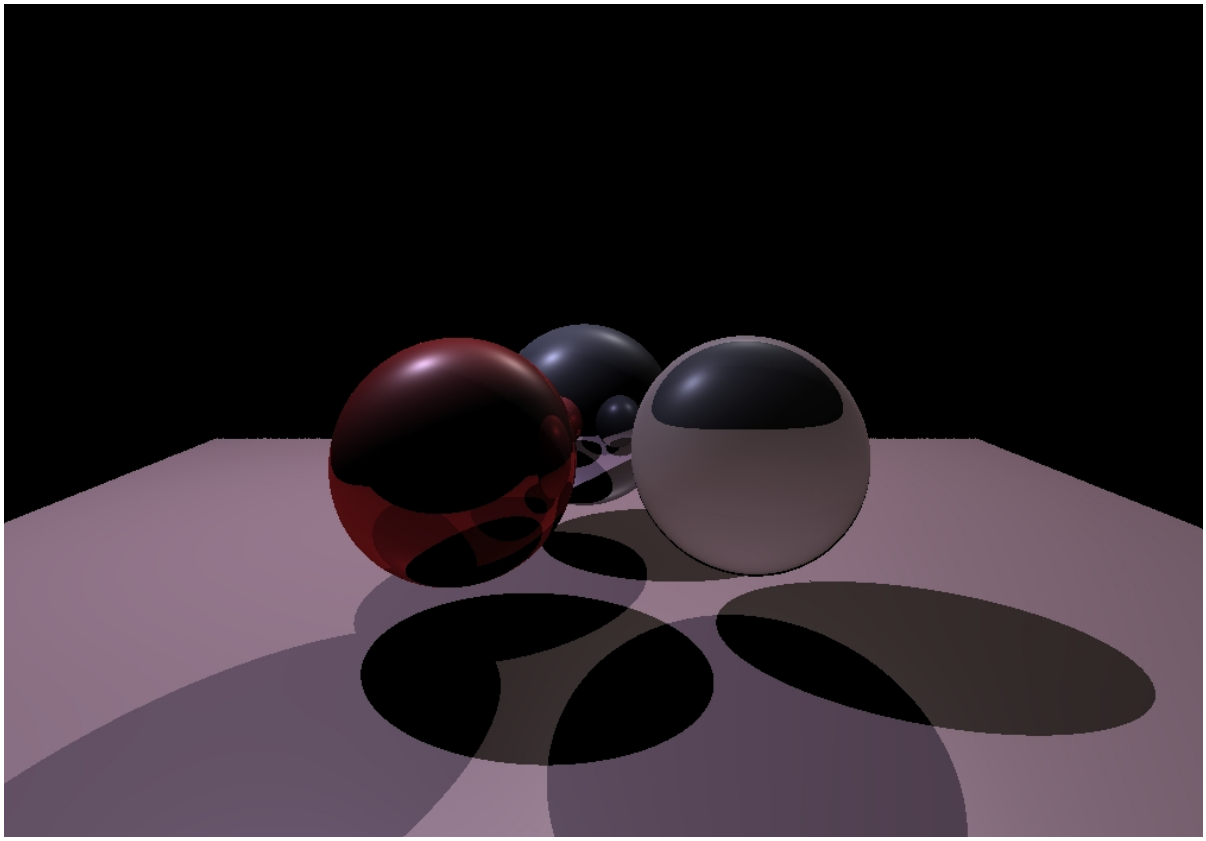


Obrázok 4.1 Zníženie počtu vrhaných lúčov, lúče sa vrhajú iba v bielych bodoch

4.1.3 Scéna

Celá scéna je realizovaná triedou *Scene*, ktorá obsahuje ukazovatele na jednotlivé objekty scény. Scéna sa inicializuje volaním metódy *InitScene*. V tejto metóde sú vytvorené všetky objekty scény vrátane svetiel. Scéna sa skladá z jednotlivých základných objektov, ako sú gule a roviny. Tieto objekty sú reprezentované triedou *Primitive*. Každá vytvorená guľa musí mať definovaný stred v premennej a_Center, a polomer *a_Radius*. Pre jednotlivé objekty sú definované metódy, ktoré počítajú priesečníky týchto objektov s lúčmi. Objekty musia mať taktiež definovaný materiál, ktorý reprezentuje farbu a vlastnosti povrchu objektu. Na to slúži trieda *Material*, ktorá umožňuje ukladať informáciu o farbe daného materiálu, ďalej informácie o jeho jednotlivých svetelných vlastnostiach, potrebných pre určenie osvetlenia objektu. Ďalej scéna obsahuje definíciu svetiel. Svetlá reprezentuje trieda *Light* a obsahuje informácie o polohe daného svetla a o jeho farbe. Scéna obsahuje funkcie pre každý typ objektu a to nájdenie priesečníka, vrátenie farby objekt a vrátenie normálového vektoru objektu. Celá výsledná scéna je uložená v poli pixelov, ktoré obsahujú informáciu o farbe daného bodu.

Výsledná scéna obsahuje tri rotujúce gule, pričom každá má iné povrchové vlastnosti, aby demonštrovala schopnosti techniky sledovania lúčov. Jedna z nich má celý povrch vysoko reflexný, podobný zrkadlu, na ktorom dochádza k odrazom svetla . Druhá je priehľadná a pri prechode svetla cez ňu dochádza k lomu. Tretia je červená s mierne lesklým povrchom. Ďalej sú v scéne dve svetlá a rovina na ktorú je vrhaný tieň. Ukážka scény je na obrázku 4.2.



Obrázok 4.2 Ukážka vytvorenej scény

4.1.4 Pohyb založený na čase

Pohyb v intre je realizovaný prostredníctvom násobenia smerového vektoru objektu nejakou konštantou, alebo iným vektorom. To má za následok, že pri väčšej zobrazovacej frekvencii je tento pohyb rýchlejší, naopak pri pomalšej frekvencii pomalší. Toto rieši metóda pohybu založenom na čase. Pri pohybovaní objektov v scéne, je tento pohyb ešte násobený časovou konštantou. Je to väčšinou čas, ktorý uplynul od posledného zobrazenia. To rieši problém s rýchlosťou pohybu a tá už nezáleží na zobrazovacej frekvencii, ale pohyb je riadený časom, čo znamená že určitú vzdialenosť prejde objekt vždy za rovnaký čas a je jedno či sa za tento čas obraz obnoví 10 alebo 100krát.

4.1.5 Zobrazenie scény

Celá scéna je uložená v poli o rozmeroch x*y, kde x je šírka obrazu a y je jeho výška. To znamená, že pre každý bod obrazu je v poli informácia o jeho farbe. Táto informácia je uložená vo farebnom formáte RGB. Na samotné zobrazenie scény je využitá knižnica windows.h, a funkcie knižnice WinAPI. Vytvorí sa okno zadanej veľkosti, do ktorého sa zobrazí obsah bufferu, obsahujúci informácie o farbe pre každý pixel obrazu.

4.2 Veľkosť výsledného súboru

Vytvoriť program, ktorý niečo zmysluplné robí a pritom je jeho veľkosť menšia ako 64kB, je pri použití normálnych postupov takmer nemožné. Veľkosť vytvoreného spustiteľného súboru ďaleko presahovala zadaných 64kB. Existujú však ale programy na kompresiu binárnych súborov. Tieto programy pracujú na princípe, že zo súboru odoberajú nepotrebné dáta. Ďalej tento súbor zabalia a pridajú do aplikácie samorozbaľovací kód, ktorý aplikáciu rozbalí do pamäti, odkiaľ je následne spustená. To všetko s nulovým oneskorením, takže po spustení aplikácia okamžite beží a netreba čakať na jej rozbalenie. Existujú desiatky týchto nástrojov. Na výsledný program som aplikoval viaceré, aby som zistil, ktorý poskytuje najväčšiu kompresiu dát. Ich zoznam a výsledky sú zobrazené v tabuľke 4.1.

Program kkrunchy z týchto nástrojov dosahuje najlepšie výsledky, a je aj najpoužívanejší kompresným programom na kompresiu grafických intier. Je to preto, že bol vytvorený práve za účelom znižovania veľkosti grafických intier a bol vytvorený samotnými tvorcami intier. Poskytuje výslednú kompresiu až okolo 80%, čo znamená, že spakovaný súbor bude mať 20% veľkosti pôvodného súboru. Program UPX za ním o niečo zaostáva, ale stále poskytuje výbornú kompresiu približne 75-70%. Obidva tieto programy sú voľne dostupné na internete na stránkach [10] a [11]. Poskytujú výborný kompresný pomer pri zachovaní dobrej rýchlosti.

Ďalším nástrojom je Strip, ktorý je priamo súčasťou programátorských nástrojov, v tomto prípade Visual Studio 2008. Ten síce neposkytuje veľkú kompresiu, ale s použitím s niektorým z vyššie uvedených kompresných nástrojov sa dá dosiahnuť veľmi dobrý výsledok, a malá veľkosť výsledného súboru.

Nemenej dôležité pre minimálnu veľkosť výsledného súboru je aj použitie správnych parametrov pri preklade zdrojových súborov. Sú to hlavne parametre /Os (minimal size) a /O1 (Favor Small Code), ktoré zaručia, že sa pri preklade do strojového kódu použijú konštrukcie, ktoré zaberajú najmenej miesta.

Pôvodná veľkosť	Program	Výsledná veľkosť	Kompresia
226816 B	UPX	108 032 B	52,4 %
226816 B	kkrunchy	83 968 B	63 %
226816 B	Strip	115 200 B	31,1 %
226816 B	Strip + UPX	54 272 B	76,1 %
226816 B	Strip + kkrunchy	48 640 B	80,4 %

Tabuľka 4.1 Porovnanie pakovacích programov

Nakoniec som použil kombináciu nástroja Strip a pakovacieho programu kkrunchy, pretože poskytovali najväčšiu kompresiu súboru. S použitím týchto nástrojov a správnym nastavením prekladača sa mi podarilo získať výsledný súbor s veľkosťou 48 640 B (48,6 kB).

5 Možnosti pokračovania na projekte

Možnosti ďalšieho pokračovania sú široké, pretože vytvorené intro určite nie je vo všetkých smeroch dokonalé. Či už je to zvyšovanie kvality zobrazenia, rýchlosti zobrazenia alebo rozšírenie jednotlivých scén a dĺžky trvania intra, možností pre pokračovanie v práci je veľa.

Intro je veľmi jednoduché, a kvalitou a rozsahom ďaleko zaostáva za profesionálnymi intrami, ktoré tvoria niektorí členovia demoscény. Tieto intrá obsahujú prepracovanejšie scény, ktoré trvajú omnoho dlhšie.

Preto by sa vytvorené intro mohlo rozšíriť o viacej rôznych scén, ktoré by sa striedali a obsahovali viac objektov. To by však ale viedlo k zväčšeniu veľkosti výsledného programu, takže by sa museli optimalizovať jednotlivé techniky, aby vo výslednom súbore nezaberali toľko miesta. Zvýšenie počtu objektov v scéne môže mať negatívny vplyv na výkon intra a môže spôsobiť, že sa rýchlosť celkového intra ešte viac zníži. To by vyriešilo naimplementovanie ďalších urýchľovacích techník, ktoré urýchľujú výpočet priesečníka lúčov s objektmi v scéne, alebo znižujú počet sledovaných lúčov. Aj technika vyhladzovania, ktorá je v programe implementovaná iba veľmi jednoduchým spôsobom, by sa dala ešte viac optimalizovať a rozšíriť, čím by sa znížil počet počítaných lúčov o viac ako polovicu.

Taktiež je možné do programu zahrnúť možnosť zobrazovania ďalších geometrických objektov a ich rôzne kombinácie (technika CSG). V intre sú zobrazované iba jednofarebné objekty, preto by ďalším vylepšením mohlo byť nanesenie textúr na objekty, čo by zvýšilo celkovú vizuálnu stránku intra.

Ďalším možným rozšírením by mohlo byť vloženie hudby do intra, pretože hudba je dôležitou súčasťou každého intra. Väčšina intier vytvorených na profesionálnej úrovni obsahuje nejaký hudobný sprievod. Zvyčajne ide o jednoduché hudobné ukážky a pohyb objektov v intre je zosynchronizovaný s touto hudbou.

6 Záver

Cieľom práce bolo vytvoriť grafické intro využívajúce techniku sledovania lúčov, ktorého veľkosť nepresiahne 64kB. Všetky tieto požiadavky sa podarilo splniť. Najprv som sa musel bližšie oboznámiť s technikami používanými pri tvorbe grafických intier a metódy sledovania lúčov, pretože som dovtedy veľa o týchto oblastiach nevedel. Až potom bolo možné začať so samotnou implementáciou výsledného programu. Výsledkom je program, ktorý po spustení zobrazí krátku animáciu, demonštrujúcu použité techniky.

Hlavným problémom, ktorý sa vyskytol počas tvorby intra, bolo dodržanie obmedzenia na 64kB. Tento limit mohol byť splnený iba s použitím špeciálnych programov, ktoré znižujú veľkosť výsledného súboru. Nakoniec sa podarilo vytvoriť program ktorého veľkosť je 48,6 kB.

Ide o môj prvý väčší projekt v oblasti počítačovej grafiky, preto výsledné intro svojou kvalitou a spracovaním nedosahuje kvalitu intier, vytvorených profesionálnymi členmi demoscény, ktorý sa tejto oblasti počítačovej grafiky venujú dlhšiu dobu. Výsledok však splňuje účel za ktorým bol vytvorený.

Za hlavný prínos mojej práce považujem to, že som získal veľa skúseností z odboru programovania a počítačovej grafiky. Taktiež som sa zoznámil s rôznymi technikami tvorby grafických intier a metódami používanými technikou sledovania lúčov.

Vytvorený program je plnohodnotný ray tracer, preto je možné jeho použitie aj v iných projektoch a prácach, nadväzujúcich na tento projekt. Je veľa možností ďalšieho pokračovania v práci na projekte a tieto možnosti sú popísane v piatej kapitole.

Literatúra

[1] Žára J., Beneš B., Sochor J., Felkel P.: Moderní počítačová grafika, Computer Press, Brno 2004

[2] Shirley P., Morley R.: Realistic Ray Tracing, 2nd edition, A K Peters, Ltd., Natick 2003

[3] Glassner A., An Introduction to Ray Tracing, Morgan Kaufmann Publishers, Inc., San Francisco 2002

[4] Chaudhuri Siddhartha: Raytracing Reference [online], 2002, Dostupné na URL http://fuzzyphoton.tripod.com/rtref/rtref_a.htm

[5] Bikker Jacco, Raytracing Topics & Techniques [online], 2004, Dostupné na URL http://www.flipcode.com/archives/Raytracing_Topics_Techniques-Part_1_Introduction.shtml

[6] Codermind team, A raytracer in C++ [online], Dostupné na URL http://www.codermind.com/articles/Raytracer-in-C++-Part-I-First-rays.html

[7] CS demoscene portal scene.cz [online], Dostupné na URL http://www.scene.cz

[8] Ray tracing (graphics) [online], Dostupné na URL http://en.wikipedia.org/wiki/Backwards_ray_tracing

[9] Phong shading [online], Dostupné na URL http://en.wikipedia.org/wiki/Phong_shading

[10] kkrunchy, pakovací program [online], Dostupné na URL http://www.farbrausch.de/~fg/kkrunchy

[11] UPX, pakovací program [online], Dostupné na URL http://upx.sourceforge.net/

Zoznam príloh

Príloha 1. CD s programom, zdrojovými súbormi, prácou v elektronickej podobe a plagátom pre prezentáciu projektu