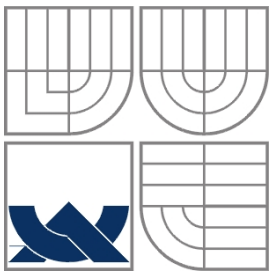
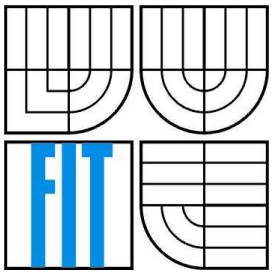


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



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

64KB GRAPHICS INTRO BASED ON RAY-TRACING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL BUTKO

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JIŘÍ HAVEL

BRNO 2010

Abstrakt

Tato práce pojednává o grafických intrech s omezenou velikostí a o technikách založených na principu sledování paprsku. Slouží jako teoretický úvod do problematiky, ze kterého následně přechází ke popisu návrhu a implementace grafického intra s obmezenou velikostí spustitelné aplikace na 64kB a využívající metodu sledování paprsku. Práce taky shrnuje získané poznatky, závěry zhodnocení výsledné aplikace. Proto může tento text sloužit pro případné studijní účely nebo jako pomoc při řešení stejně zaměřených úkolů.

Abstract

This thesis is concerning about graphic intro with limited size and about techniques based on ray tracing principles. The thesis serves as theoretical introduction and description of a concept and implementation of a graphic intro with limited size to 64kB and based on ray tracing. Acquired informations and resume of created application are mentioned as well. Because of that, this text can serve for eventually study purposes or help with finding answers for similar problems.

Klíčová slova

grafické intro s obmezenou velikostí, demoscéna, sledování paprsku v reálném čase, akcelerační metody, Phong, Cook-Torrance.

Keywords

graphic intro with limited size, demoscene, real-time ray tracing, accelerating techniques, Phong, Cook-Torrance.

Citace

Butko Michal: Grafické intro 64kB s použitím sledování paprsku, bakalářská práce, Brno, FIT VUT v Brně, 2010

Grafické intro 64kB s použitím sledování paprsku

Prohlášení

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

……
Michal Butko
14.05.2010

Poděkování

Na tomto místě bych rád poděkoval Ing. Jiřímu Havlovi za jeho vstřícný přístup, ochotu a věcné připomínky.

Obsah

1 Úvod

Súčasťou snahy o realistické grafické zobrazovanie a vizuálne úchvatné efekty je neustály boj medzi výpočtovou náročnosťou a ich vizuálnym výsledkom. Práve výpočtová náročnosť bol faktor, ktorý zabraňoval fotorealistickým technikám založeným na princípe sledovania lúčov, aby sa uplatnili v praktickom použití. Tieto sa až donedávna tešili záujmu výhradne zo strany vedeckých kruhov a významnejšieho komerčného užitia sa dočkali až nedávno. Toto dlhoročné dospievanie *ray tracing* sprevádza už odo dňa svojho zrodenia do sveta počítačovej grafiky. Aj keď bola táto metóda predstavená už v roku 1968 a svojho času zbudila veľký rozruch v odborných kruhoch, spracovania v reálnom čase, sa dočkala až o 18 rokov neskôr príchodom výkonnejšej výpočtovej techniky.

Spolu s nástupom nových technológií, v tom čase ešte veľmi obmedzených, sa realistickým metódam otvorili nové možnosti. Metódy sledovania lúčov postupne dorastali do stavu, kedy prestávali byť zaujímavé výhradne pre vedecké kruhy, ale stávali sa prínosné napríklad aj pre vznikajúcu komunitu nadšencov okolo tvorby grafických intier. Technická náročnosť pri vývoji grafických aplikácií pracujúcich v reálnom čase na tomto princípe, bola výzva, ktorá priťahovala široké spektrum programátorov. Spolu s možnosťou využitia efektov jedinečných pre tento spôsob grafického zobrazenia, tak vznikol fenomén, ktorý sa teší obľube doteraz.

A práve tvorbe takého grafického intra sa venuje aj táto práca, ktorej cieľom je vytvoriť neinteraktívnu grafickú aplikáciu s obmedzenou veľkosťou, nepresahujúcou 64kB, pracujúcu na princípe sledovania lúčov v reálnom čase. Výsledná animácia musí predovšetkým pútavým spôsobom demonštrovať grafické možnosti vytvoreného nástroja. Vzhľadom na spomínanú výpočtovú náročnosť je dôležité dosiahnuť maximálnej rýchlosti vykresľovania, a to bohužiaľ čiastočne aj na úkor výsledného zobrazenia a zložitosti scény. No napriek zásadným obmedzeniam čo do počtu objektov v animácii je možné pri správnom a kreatívnom prístupe dosiahnuť esteticky pútavej scény.

Nasledujúci text bude preto pojednávať o aspektoch spojených s danou problematikou, bude popisovať postupy aké som volil pri vývoji. Bude popisovať prekážky, ktoré bolo potrebné prekonať v jednotlivých etapách, kde budem následne predkladať ich riešenie.

Samotnú kapitolu rozoberajúcu návrh a implementáciu zadania, predchádza teoretický úvod do problematiky. Jedná sa o kapitolu venujúcu sa primárne základom nezbytným pre pochopenie problematiky, ale dotýka sa čiastočne aj tematicky zaujímavých stránok úlohy.

Na záver úvodu musím upozorniť, že text sa implicitne venuje rýchlejšiemu ,a teda použiteľnejšiemu spätnému sledovaniu lúčov.

2 Grafické demo

V prvej a kratšej teoretickej kapitole, sa budem venovať intrám s obmedzenou veľkosťou. Táto časť nie je čo sa samotnej implementácie týka nezbytne nutná, považujem ju ale za prínosnú pre pochopenie účelnosti zadania práce a jej korene v reálnom svete.

2.1 Intro

Grafické intro je neinteraktívna multimediálna prezentácia, ktorej hlavným úmyslom je demonštrácia schopností jej autorov. Spája v sebe ako technické zručnosti tvorcov, tak ich umelecké nadanie a kreatívne myslenie. Samotný termín *intro*, bol medzi ich tvorcami chápaný, ako nekonečné demo, používané crackermi za účelom podpisu softwaru u ktorého úspešne prelomili ochranu. Postupne sa scéna okolo tvorby grafických animácii menila a menil sa aj význam slova intro. Dnes sa pod týmto pojmom rozumie prevažne grafické demo, ktoré je ľubovoľným spôsobom veľkostne obmedzené. Demá zastrešujú široké spektrum rôznych kategórií, ktoré ale už nie sú pre naše účely príliš zaujímavé.

Teda intrá, ako veľkostne obmedzené animácie, tvoria z technického hľadiska náročnejšiu scénu tvorby. Narozdiel od pamäťovo neobmedzených projektov, ktoré sú často-krát produkované s využitím softwaru s vysokou mierou abstrakcie a notnou dávkou automatizácie, u intier technická stránka a technické znalosti a schopnosti tvorcov pri vývoji dominuje. Preto je často-krát vyžadované množstvo inovácií, vynikajúca znalosť operačného systému a veľa ďalších programátorských cností. V súčasnosti nato aby intro naozaj zaujalo a dostalo sa medzi špičku, musí vývoj prebiehať ako tímový projekt. Kde produkcia jednotlivých celkov často-krát prebieha oddelene. Čo platí hlavne pre ozvučenie intra, ktoré je samostatnou rozsiahlou súčasťou grafických intier.

Hlavne z dôvodu všeobecnej súťaživosti medzi tvorcami, ale aj tímovej spolupráce začali časom vznikať komunity vývojárov zamerané na demo produkciu nazývané *demoscény*.

2.2 História

Prvé programy, ktoré niesli znaky grafických intier sa objavili už začiatkom 50tých rokov. Jednalo sa o takzvané *display hack*, ktoré na obrazovku vykresľovali jednoduché pohybujúce sa obrazy. Prezentácie plne kvalifikovateľné ako demá, začali vznikať, až na prelome 70tých a 80tých rokov. Ich vznik je spjatý so zrodom počítačového pirátstva. Kedy skupina, nazývajúca sa *crackeri*, prelamovali ochranu softwaru. Aby noví majitelia kopírovaného softwaru vedeli komu za prelomenie ochrany vďačia začali crackeri vkladať do programov grafické intrá ako svoj elektronický podpis, ktorým prezentovali kto ochranu prelomil.

Pre prvé animácie boli z pochopiteľných dôvodov typické drastické pamäťové omezenia, s veľmi skromnými výpočtovými možnostiami. Z počiatku sa často jednalo len o statické obrazy s pohyblivými titulkami. Postupne ako vznikala medzi *crackermi* rivalita, čo do počtu prelomených ochrán a čo do kvality vytvorených intier. Viedla ich k stále impresívnejším a náročnejším dielam, kedy intro už neslúžilo len ako značka, ale demonštrovalo aj schopnosti jeho tvorcu. Sparťanské podmienky hardwarového vybavenia tej doby na nich kládli vysoké nároky na pamäťovú úspornosť a nútili ich premýšľať ako svoje techniky vylepšovať a hľadať triky, ktorými by dosiahli požadovaných efektov.

Obrázok 2.1 - http://ultraforce.com - Vectordemo (1991)

Jedno z prvých 3D dém na PC

Naviazanosť na počítačové pirátstvo netrvala príliš dlho a ako sa okruh ľudí okolo demoscény postupne rozrastal a začali sa objavovať nadšenci, ktorí už tvorili samostatné demá, bez ich využívania v prelomených programoch. Tento trend ďalej pokračoval a s nástupom prvých PC začiatkom 90tých rokov sa demoscéna od pirátstva úplne dištancovala.

2.3 Demoscéna

Od počiatku bol vznik intier úzko spjatý so súperením medzi ich autormi. Tento aspekt pretrval až dodnes, kedy demoscéna, ako skupina nadšencov vytvárajúcich grafické či zvukové demá, existuje hlavne za účelom súťaženia. V súčasnosti sa tak konajú každoročne súťaže s dlhou tradíciou, kvalitne organizované a na profesionálnej úrovni.

Demá medzi sebou súperia v jednotlivých kategóriách. Existuje množstvo rôznych skupín a kritérií podľa, ktorých sa rozhoduje kam sa demo zaradí. K najpodstatnejšiemu deleniu, ale dochádza medzi príspevkami, na ktoré sú uplatňované kapacitné obmedzenia a príspevky bez obmedzení. Zatiaľ čo u tých druhých prevláda hodnotenie umeleckej stránky, u intier s obmedzenou veľkosťou je stále hodnotený najmä dosiahnutý technický výsledok, či použitie nových efektov a podobne. Tu je určite zaujímavý fakt je, že najdominantnejšími, medzi dielami s obmedzením, sú kategórie grafických intier do 2kB a 64kB. K ďalšiemu deleniu potom dochádza na základe platformy, formátu alebo štýlu.

Obrázok 2.2 - http://www.rgba.org/ - z kategórie 4kB

Typické demo je produkované demoskupinou, čo je tým demo-vývojárov. V súčasnosti v týchto skupinách prevláda úzka špecializácia tvorcov dém, typické delenie je na programátora, grafika a zvukára. No napriek tomu do súťaží stále prispieva veľa samostatne tvoriacich vývojárov a aj keď demoskupiny obsahujú často tucty členov, na samotnom vývoji sa málokedy podieľa viac ako desať ľudí.

Na záver je vhodné dotknúť sa prínosu tohto fenoménu, kde cez to, že umelecká stránka je stále sporná a demo tvorba nie je príjmaná ako umenie, má významný vplyv na ďalšie oblasti informatiky.

3 Sledovanie lúčov

Metóda spätného sledovania lúčov je fotorealistická metóda, založená na vrhaní lúčov z oka pozorovateľa smerom do scény. Svoj pôvod má už vo fyzike, kde ju fyzici využívali pri výpočtoch optických javov ako lom alebo odraz svetla. Preto s nástupom počítačovej grafiky a snahy o čo fyzikálne najvernejšie metódy, dostala táto metóda možnosť uplatniť sa tiež v obore informatiky. A napriek tomu, že v tej dobe šlo o extrémne náročnú fyzikálnu simuláciu, vedci správne predpokladali, že s nástupom výkonnejších počítačov, by mohla nájsť praktické využitie. A tak s príchodom výkonnejšieho hardwarového vybavenie, si táto metóda postupne začala nachádzať uplatnenie aj v bežnej praxi. Predovšetkým tam, kde bola požadovaná vysoká kvalita zobrazenia, bez nárokov na dobu vykresľovania.

Bohužiaľ *ray tracing* v reálnom čase ostával stále skôr otázkou vedeckých kruhov. Našťastie s nástupom výkonných osobných počítačov, prestáva platiť aj toto obmedzenie a *ray-tracing* v reálnom čase sa začína tešiť značnej popularite. A blýska sa na ešte lepšie časy s možným nástupom grafických kariet podporujúcich tieto techniky. Ich prípadné uvedenie na trh by mohlo znamenať zásadný zvrat, keďže súčasné grafické akcelerátory túto grafickú metódu žiadnym spôsobom nepodporujú a všetka výpočtová záťaž je tak odkázaná na procesor.

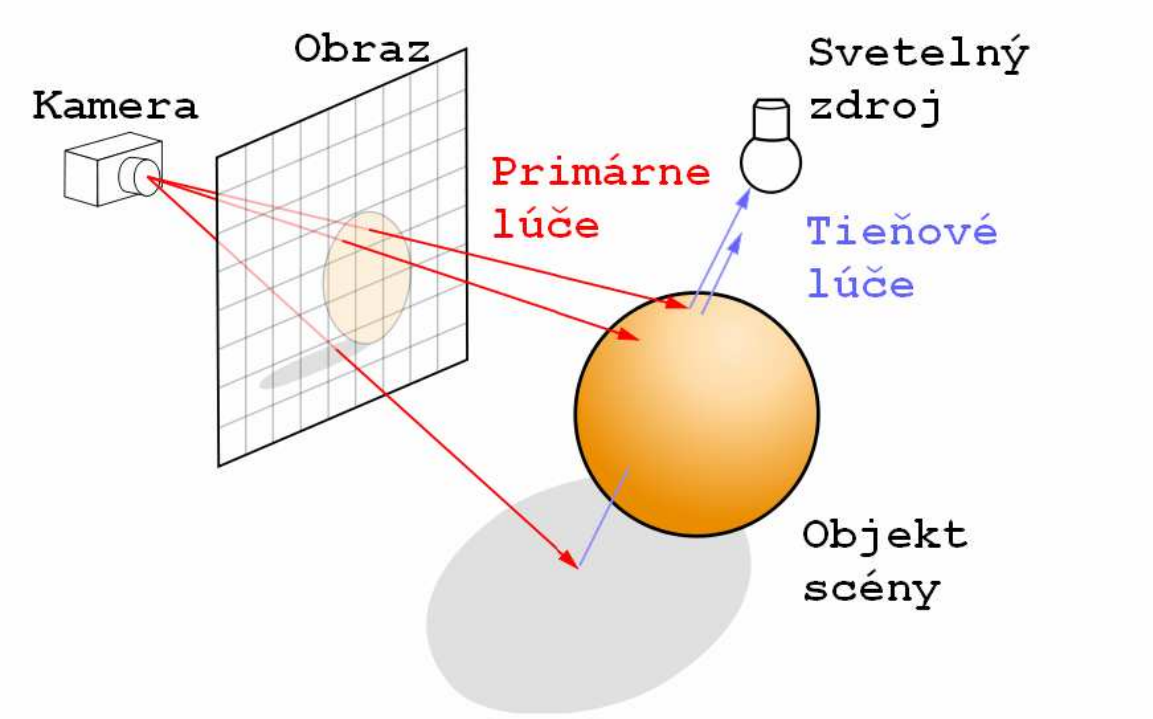
Ako bolo okrajovo spomenuté už v úvode, pojem sledovanie lúčov zahrňuje viac metód vykresľovania s podobným princípom, ale odlišným prístupom. Optike skutočného sveta je najbližší algoritmus dopredného sledovanie lúčov. Odpovedá realite, kde sú lúče vysielané rovnomerne zo zdroja svetla do okolitého sveta, tam narážajú na objekty, na ktorých je časť ich fotónov pohltená, odrazená alebo prepustená skrz ďalej do priestoru. Zo všetkých lúčov, ktoré takto dopadli na jeden bod sietnice nášho oka, či už sekundárnych alebo priamych, je sčítaná ich farebná zložka, ktorá odpovedá jednému pixelu v rastrovom zobrazení monitora.

Tento algoritmus je, ale ako grafická simulácia neúnosne náročný. Z každého svetelného zdroja musí byť vyslaný obrovský počet lúčov, z ktorých len zanedbateľné percento zasiahne oko pozorovateľa alebo v našom prípade priemetňu pred kamerou. Preto je pre nás oveľa zaujímavejší algoritmus spätného sledovania lúčov, ktorý narozdiel od dopredného algoritmu vysiela len lúče, ktoré budú mať prínos pre vykreslenie scény. Ako je už z názvu pravdepodobne jasné ide o opačnú techniku, kedy je lúč vyslaný do scény z oka pozorovateľa. Práve tomuto typu *ray traceru* sa bude práca ďalej venovať.

3.1 Princíp spätného sledovania lúčov

3.1.1 Sledovanie lúčov prvého rádu

Samotný algoritmus primitívneho *ray traceru* prvého rádu je relatívne jednoduchý. Z kamery, počiatočného bodu, sú lúče vrhané postupne skrz projekčné okno. Toto projekčné okno je rozdelené tak, aby mal každý pixel v okne svoj ekvivalentný bod, cez ktorý bude jeho lúč prechádzať. Postupne sa takto skrz priemetňu vystreľujú lúče pre zistenie farebnej zložky každého bodu obrazu. Lúče vedené do scény priamo z kamery sa nazývajú primárne, v reálnom svete by sa teda jednalo práve o svetelné lúče, ktoré nám nesú výslednú farebnú zložku na sietnicu.



Obrázok 3.1 – Princíp sledovania lúčov

Pre každý primárny lúč sa musia vypočítať priesečníky s telesami v scéne, kde sa za výsledný prienik uvažuje len zásah s najbližším objektom vzhľadom ku kamere. V prípade, že lúč všetky telesá minul, je na daný bod zanesená farba pozadia. V závislosti na tvare sú výpočty priesečníkov rôzne náročné, napríklad priesečník s jednoduchou rovinou je vyjadrený jednoduchým vektorovým vzťahom:

$$\vec{\mathbf{N}} \bullet (\vec{\mathbf{E}} + t\vec{\mathbf{D}} - \vec{\mathbf{Q}}) = 0$$

kde $\vec{\mathbf{N}}$ je vektor normály, $\vec{\mathbf{Q}}$ je bod na rovine, $\vec{\mathbf{D}}$ normalizovaný vektor smeru lúča a bod pohľadu $\vec{\mathbf{E}}$, vypočítaný ako rozdiel medzi počiatkom lúča so stredom gule.

Priesečník s guľou reprezentuje rovnica:

$$t^2(\vec{\mathbf{D}}\bullet\vec{\mathbf{D}}) + t(2\vec{\mathbf{E}}\bullet\vec{\mathbf{D}}) + (\vec{\mathbf{E}}\bullet\vec{\mathbf{E}}) - R^2 = 0$$

Pre výpočet priesečníku s lúčom, ktorý je daný miestom pôvodu $\vec{\mathbf{O}}$ a smerom $\vec{\mathbf{D}}$ s trojuholníkom, ktorý je určený bodom $\mathbf{A}$ a vektorom $\vec{\mathbf{b}}, \vec{\mathbf{c}}$, je potrebné vypočítať miesto prieniku s rovinou, na ktorej tento leží a to pomocou rovnice:

$$\vec{\mathbf{O}} + t\vec{\mathbf{D}} = \mathbf{A} + \beta\vec{\mathbf{c}} + \gamma\vec{\mathbf{b}}$$

Tento bod leží zároveň trojuholníku keď a iba keď platí $\beta > 0, \gamma > 0, \beta + \gamma < 1$.

Následne v prípade zásahu, je ešte nutné viesť tieňový lúč smerom k zdrojom svetla, ktorým sa zisťuje či je zasiahnutý bod niektorým z nich skutočne osvetlený. Overenie prebieha opäť testovaním priesečníkov tieňového lúča s telesami scény, ktoré by mohli bod zásahu zatieniť. V tomto prípade sa už, ale nemusí hľadať najbližší prienik a akýmkoľvek stretom s tieňovým lúčom, je príspevok daného svetelného zdroja eliminovaný. Následné je možné s využitím vybranej osvetľovacej metódy a vlastností povrchu zasiahnutého objektu dopočítať výsledné zafarbenie počítaného bodu.

3.1.2 Sledovanie lúčov vyšších rádov

V reálnom svete dochádza k úplnému pohlteniu lúčov málokedy, práve naopak sa fotóny od povrchov telies odrážajú alebo nimi dokonca prechádzajú. Pre dosiahnutie týchto opticky zaujímavých efektov, pre ktoré je *ray-tracing*, tak obľúbený, je potrebné použiť rekurzívneho sledovania lúčov vyššieho rádu.

Po dopade primárneho lúča sa vyhodnocuje povrch, na ktorý lúč dopadol a okrem farebných alebo difúznych vlastností musí informácia obsahovať tiež odrazovosť alebo priehľadnosť materiálu spolu s jeho indexom lomu. Tieto vlastnosti rozhodujú či budú vyslané sekundárne lúče, ktoré zisťujú prírastok farebnej zložky odrazených, respektíve lomených lúčov, ktoré prenikli priehľadným telesom. Smer sekundárneho lúča sa dopočítava buď z uhlu dopadu alebo z indexu lomu telesa a jeho okolia. Konkrétne pre výpočet smeru odrazeného lúča je užitý vzťah:

$$\vec{\mathbf{R}} = \vec{\mathbf{I}} - 2(\vec{\mathbf{N}}\bullet\vec{\mathbf{I}})\vec{\mathbf{N}}$$

kde $\vec{\mathbf{R}}$ je vektor odrazeného lúča, $\vec{\mathbf{I}}$ pôvodného lúča a $\vec{\mathbf{N}}$ je normálový vektor objektu v mieste dopadu.

Pre výpočet smeru lomeného lúča je užitý vzťah:

$$\vec{\mathbf{T}} = \frac{n_1}{n_2}\vec{\mathbf{I}} - \left(\frac{n_1}{n_2}\cos\theta_{\mathbf{i}} + \sqrt{1-\sin^2\theta_{\mathbf{t}}}\right)\vec{\mathbf{N}}$$

kde $\vec{\mathbf{T}}$ je vektor smeru výsledného lomeného lúča, $\vec{\mathbf{I}}$ je vektor dopadu, $\vec{\mathbf{N}}$ je normálový vektor objektu v mieste dopadu, n_1 je index lomu pôvodného materiálu, v ktorom sa lúč pohyboval pred

prienikom a n_2 je index lomu objektu do ktorého lomený lúč vstúpil. Zatiaľ čo v rovnici uhol dopadu θ_i je známy, uhol lomu θ_t sa ďalej dopočíta pomocou Snellovho zákona:

$$\sin^2 \theta_t = (\frac{n_1}{n_2})^2 (1 - \cos^2 \theta_i)$$

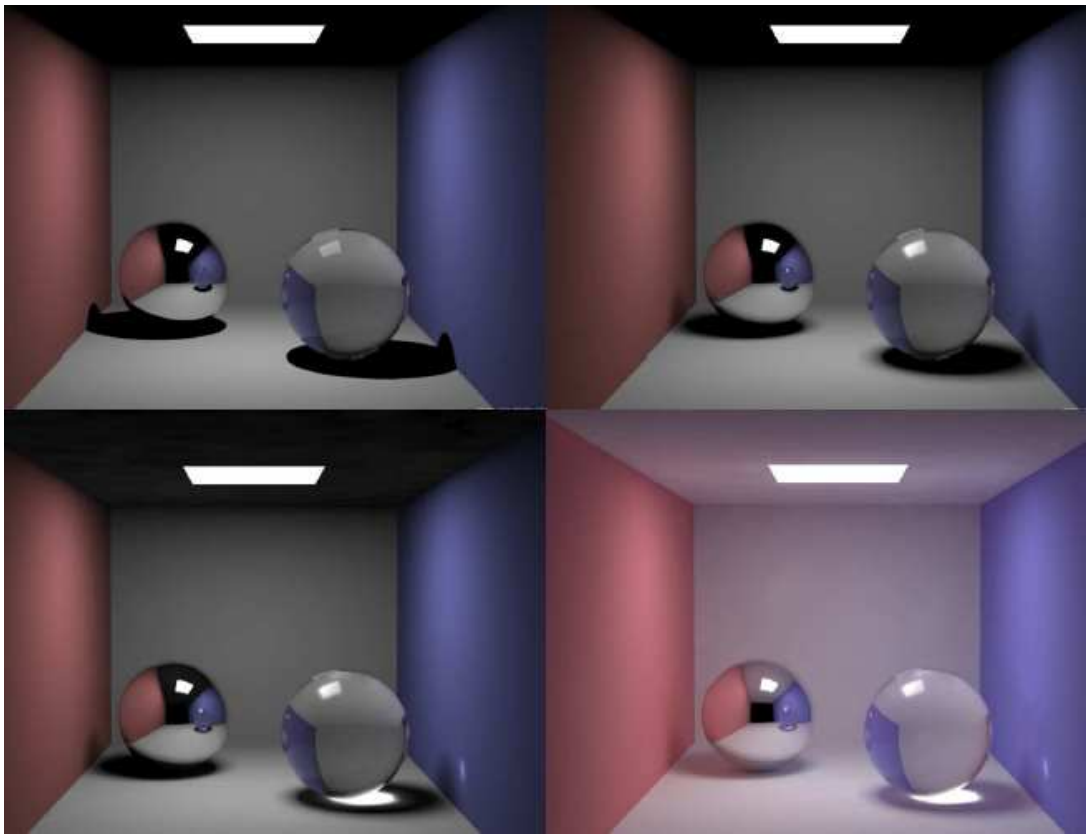
Z algoritmu vyplýva, že takýchto lúčov vyššieho rádu môže byť rekurzívne vyslaných neobmedzene mnoho, čo môže zbytočne spomaľovať výpočet bez väčšieho grafického zlepšenia. Preto je na mieste definovať maximálnu hĺbku rekurzie.

3.1.3 Algoritmus

```
Raytrace( počiatočný bod, smer lúča ){
  vzdialenosť = nekonečno;
  najbližší_objekt = NULL;
  farba = 0x0;
  for každý objekt {
    najdi t, najmänšie, nie negatívne riešenie priesečníku;
    if t exists AND t < vzdialenosť {
      vzdialenosť = t;
      najbližší_objekt = aktuálny_objekt;
    }
  }
  if najbližší_objekt == NULL {
    return farba;
  }
  for každý zdroj svetla {
    if tieňový_lúč( bod prieniku, pozícia svetla ) == osvetlené {
      farba += svetelný prírastok;
    }
  }
  if najbližší_objekt.koef_odrazivosti > 0 {
    zrkadlená_farba = Raytrace( bod prieniku, odrazený vektor )
    farba += koef_odrazivosti * zrkadlená_farba ;
  }
  if najbližší_objekt.koef_priehľadnosti > 0 {
    presvitajúca_farba = Raytrace( bod prieniku, lomený vektor )
    farba += koef_priehľadnosti * presvitajúca_farba ;
  }
 return farba;
}
```

3.2 Optické nedostatky

Napriek tomu, že sledovanie lúčov je technika významná pre jej fotorealičnosť, existujú javy s ktorými si jej typická implementácia nevie poradiť. Jedná sa najmä o neschopnosť zobrazovať mäkké tiene, čo pramení zo skutočnosti, že pri výpočtoch sa pracuje len s bodovým svetlom, a tak tiene prechádzajú voči konkrétnemu zdroju svetla zo stavu absolútneho zatienenia do stavu plného osvetlenia. Existujú možnosti ako tento neduh odstrániť, ale ich použitie je výkonovo značne náročné a preto často-krát nevhodné. Ďalšou anomáliou odchyľujúcou sa od fotorealičnosti je neschopnosť lesklých predmetov slúžiť ako zdroje svetla, tie síce zrkadlia objekty okolo, ale už neprispievajú k ďalšiemu osvetľovaniu scény. Posledným významným neduhom typickým pre jednoduché *ray tracery* je, že pri sledovaní tieňových lúčov priehľadné telesá tvoria absolútne prekážky, ktoré neumožňujú prienik osvetľovacích lúčov.



Obrázok 3.2 - Optické vlastnosti, ktoré klasické sledovanie lúčov nerieši [7]

Napriek tomu, sú techniky ako tieto nedostatky odstrániť, bohužiaľ väčšinou za cenu menšieho alebo väčšieho nárastu časovej náročnosti výslednej aplikácie.

3.3 Osvetľovacie modely

Osvetľovací model definuje systém rozdeľovania svetelných prírastkov od jednotlivých svetelných zdrojov. Na tento účel existujú rôzne techniky, ktoré poskytujú vývojárovi rozdielne vizuálne efekty a výpočtovú náročnosť. Obecne platí, že sa jedná vždy o kompromis medzi náročnosťou modelu a mierou reálnosti výsledného obrazu.

Práve pre účely *ray traceru* je nutné sa zaoberať týmto kompromisom dvoj násobne, skrz snahu o získanie maximálnej dostupnej rýchlosti a zároveň obstojného vizuálneho výsledku. Pre tieto účely sa ako optimálny ukázal Phongov model osvetlenia, ktorý poskytuje výpočtovo prijateľný systém získavania komplexného svetelného prírastku spolu s prijateľným vizuálnym výsledkom. V súčasnosti je tento model pre metódy sledovania lúčov typický a hojne využívaný. Najmä u jednoduchých nepríliš rozsiahlych *ray-tracerov* som sa stretával výhradne s týmto modelom. Preto som sa v návrhu rozhodol použiť Phongov osvetľovací model ako primárny.

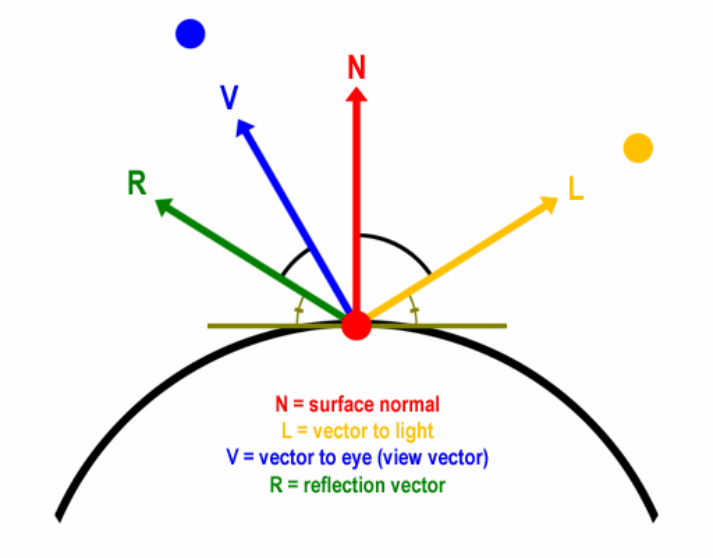
Napriek tomu trpí problémom a tým je spôsob akým dochádza k výpočtom odrazeného svetla, ktorý môže u objektov pôsobiť strojene a umelo. V tomto ohľade existujú reálnejšie a komplexnejšie techniky výpočtov pre túto časť osvetlenia a jednou z nich je Cook-Torrancov model, ktorý som použil ako možné rozšírenie implementácie. Tento model je primárne vytvorený pre drsné povrchy a najmä na modelovanie kovových a plastových materiálov. Svojim prístupom poskytuje širšiu škálu možných nastavení svetelného systému a samotných povrchových vlastností objektu, čím slúži ako komplexnejší prostriedok na rôzne vizuálne efekty. Ako sa dalo predpokladať cenou za reálnejší prístup je jeho o niečo vyššia výpočtová náročnosť.

3.3.1 Phongov osvetľovací model

Empirický osvetľovací model, ktorú v roku 1977 navrhol Bui-Tuong Phong. Model vychádza zo znalosti normálových vektorov osvetľovanej plochy $\vec{\mathbf{N}}$, smeru pohľadu $\vec{\mathbf{V}}$, smeru odrazeného lúča $\vec{\mathbf{R}}$ a vektoru ku svetelnému zdroju $\vec{\mathbf{L}}$. Výsledné osvetlenie bodu je súčtom jednotlivých osvetľovacích zložiek, a to zložkou odrazovou, difúznou a ambientnou. Zrkadlová zložka je získavaná vzťahom:

$$I_S = I_L r_S (\vec{\mathbf{V}} \bullet \vec{\mathbf{R}})^h$$

kde I_L predstavuje farebné zloženie dopadajúceho svetla, r_S je odrazová schopnosť povrchu telesa, vektor $\vec{\mathbf{V}}$ predstavuje smer pohľadu, vektor $\vec{\mathbf{R}}$ je smer ideálne odrazeného lúča a koeficient h udáva ostrosť zrkadlového odrazu.

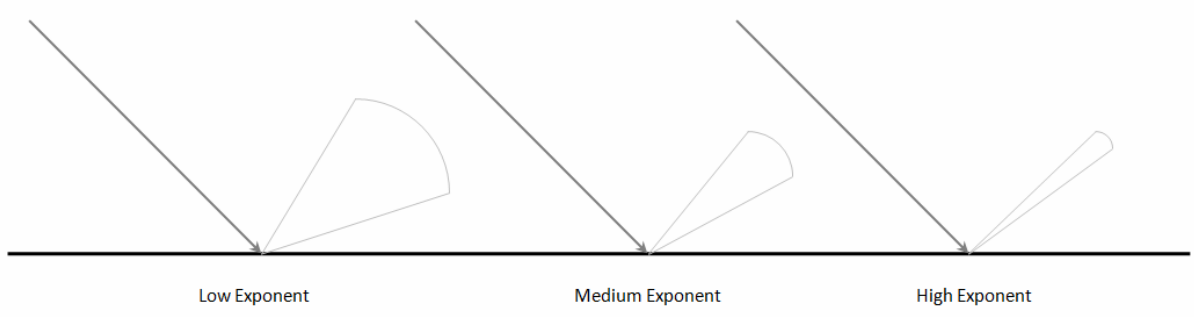


Obrázok 3.2 – Phongov osvetľovací model

Rovnica pre výpočet difúznej zložky osvetlenia je:

$$I_D = I_L r_D (\vec{\mathbf{L}} \bullet \vec{\mathbf{N}})$$

Kde r_D je stupeň zastúpenia difúzneho osvetlenia telesa a kde vzájomný vzťah, medzi vektorom smeru dopadu lúča a normálou telesa v danom bode, $\vec{\mathbf{L}} \bullet \vec{\mathbf{N}}$ poskytuje jeho natočenie voči svetlu v danom bode. Nakoniec je pripočítaná ambientná zložka osvetlenia, ktorá predstavuje okolité všade prítomné svetlo a je po celom povrchu telesa konštantná.



Obrázok 3.3 – Ostrosť zrkadlového odrazu

3.3.2 Cook–Torrancov osvetľovací model

Oproti predchádzajúcemu modelu je pre Cook-Torrancov model kľúčový spôsob výpočtu zrkadlovej časti osvetlenia, ktorá je o poznanie zložitejšia. Zatiaľ čo Phongov model pracuje pri výpočtoch zrkadlovej časti na jednoduchom matematickom modeli a poskytuje konštantnú farbu bez ohľadu na uhoľ pohľadu, tento systém prichádza s komplexnejšou technikou založenou viac na fyzike ako čisto matematickom prístupe.

Obrázok 3.4 – Cook-Torrancov osvetľovací model

Samotná odrazová zložka je tu vyhodnotená výrazom:

$$R_S = \frac{Fresnel \times Hrubosť \times Geometria}{(\vec{\mathbf{N}} \bullet \vec{\mathbf{V}}) \times (\vec{\mathbf{N}} \bullet \vec{\mathbf{L}})}$$

Plný výpočet Fresnelovej rovnice, je príliš drahý a komplikovaný, je založený na fyzikálnom princípe a počíta s parametrami ako vlnová dĺžka svetla a teplota. Preto sa pre prácu v reálnom čase skôr používa jej aproximácia predstavená Christophom Schlickom:

$$Fresnel = F_O + (1 - (\vec{\mathbf{H}} \bullet \vec{\mathbf{V}}))^5 \times (1 - F_O)$$

kde $\vec{\mathbf{H}}$ je polovičný vektor medzi smerom pohľadu a svetelným zdrojom a F_O Fresnelova obrazivosť. Pre modelovanie hrubosti sa u aplikácii v reálnom čase používa nepríliš presná Gaussova distribúcia:

$$Hrubosť = c \times e^{-(\frac{\alpha}{m^2})}$$

kde α je uhol medzi normálovým vektorom a polovičným vektorom a m je vlastná hrubosť povrchu. Nevýhodou tohto výpočtu je fakt, že je závislý na konštante c, ktorá musí byť empiricky definovaná pre najlepší výsledok. Nakoniec rovnica pre výpočet geometrie je definovaná ako:

$$Geometria = \min(1, \frac{2(\vec{\mathbf{H}} \bullet \vec{\mathbf{N}})(\vec{\mathbf{V}} \bullet \vec{\mathbf{N}})}{\vec{\mathbf{V}} \bullet \vec{\mathbf{H}}}, \frac{2(\vec{\mathbf{H}} \bullet \vec{\mathbf{N}})(\vec{\mathbf{L}} \bullet \vec{\mathbf{N}})}{\vec{\mathbf{V}} \bullet \vec{\mathbf{H}}})$$

3.4 Akceleračné techniky

Je logické, že práve technikám urýchľujúcim *ray tracer*, je pri vývoji venovaná veľká dávka pozornosti. Urýchlenie zobrazenia je dosahované predovšetkým dvomi obecnými smermi. Prvý dosahuje výsledku zrýchľovaním výpočtov priesečníkov, respektíve buď ich zefektívňovaním alebo ich minimalizáciou. Druhý smer akcelerácií sa zameriava na znižovanie počtu sledovaných lúčov. Každý prístup prináša vývojárom rozdielne podmienky pre ich použitie. Ako som mohol sám poznať pri testovaní jednotlivých metód.

V prípade *ray traceru* pre prehrávanie intra, som sa musel držať jednoduchých scén, pre ktoré boli techniky minimalizovania počtu výpočtov priesečníkov poväčšinou nevhodné. Ich prínos sa prejavuje až pri rozsiahlych scénach s vysokým počtom zložitých objektov. Naopak pre pár jednoduchých objektov, kvôli ich nevyužitej réžii, prinášajú len ďalšie spomalenie. O niečo lepšie obstoja niektoré techniky pre zefektívnenie výpočtu priesečníkov. Nakoniec techniky na znižovanie počtu priesečníkov majú v tomto smere univerzálnejšie uplatnenie, bohužiaľ občas za cenu zníženej kvality výsledného obrazu.

3.4.1 Akcelerovanie výpočtov

Tento typ techník je možné ďalej rozdeliť na dve podskupiny. Prvou sú techniky urýchľujúce výpočet priesečníkov. Jedná sa prevažne o špeciálne funkcie, ktoré nám umožňujú zistiť či bol daný objekt zasiahnutý alebo nie. Tým je väčšina objektov eliminovaná bez nutnosti pokračovania vo výpočtoch k zisteniu bodu prieniku. Tiež sem patrí metóda obálok objektov, kde sa zložité telesá obaľujú do väčších obalových telies, ktoré sú jednoduché na výpočet priesečníku. Čím zisťovanie zasiahol-nezasiahol prebieha napríklad voči gule a až po zasiahnutí obalového telesa, dochádza k výpočtu priesečníku voči zložitejšiemu objektu.

Druhou podskupinou sú techniky znižujúce počet počítaných priesečníkov, ktoré pre rozsiahle scény znamenajú zásadné zrýchlenie. Sem sa radia rôzne priestorové hierarchie, ako hierarchia obálok. Tá opäť obaľuje skupiny objektov, do telies, u ktorých sa ľahko počíta priesečník. Na rozdiel od jednoduchej techniky obálok, táto metóda obaľuje viacej obalových telies ďalšími rodičovskými objektmi a takto pokračuje dokiaľ nevytvorí hierarchickú štruktúru. Technika delenia priestoru, uzatvára objekty v scéne do malého priestoru a tento ďalej delí na menšie časti, ktorým už podľa polohy prideľuje jednotlivé objekty. Pri sledovaní lúča sa potom vyhodnotí v ktorej časti scény sa nachádza, čo vo výsledku umožní testovať len objekty, ktoré v danej oblasti skutočne sú. Síce je ľahko implementovateľná, no jej nevýhodou je, že nerieši nerovnomernosť rozloženia objektov v scéne. Lepšou technikou ako delenie priestoru je napríklad technika vnorených mreží alebo zhora budované hierarchie založené na BSP, oktalových alebo KD-stromoch. Tieto rovnako delia scénu na menšie celky, pričom, ale využívajú vhodnejší spôsob delenia. Následne tvoria jednotlivým objektom

príslušnosť v týchto menších celkoch. Výhodou napríklad je, že pri prehľadávaní sekundárne respektíve tiež tieňové lúče prechádzajú stromom od posledného spracovávaného uzla.

3.4.2 Zníženie počtu sledovaných lúčov

Medzi typické techniky znižujúce počet vysielaných lúčov patrí adaptívne riadená rekurzia a adaptívne podvzorkovanie. Zatiaľ čo adaptívne riadená rekurzia je účinná až pri veľkom počte reflexných alebo priehľadných objektov, jej výhoda je v ľahkej implementácii. Účelom tejto metódy je zastaviť rekurzívne vysielanie ďalších sekundárnych lúčov v prípade, že ich farebný prírastok, by už nemal podstatný vliv na výslednú farbu bodu.

Oproti tomu adaptívne podvzorkovanie je technika, ktorá je pri jednoduchých scénach značne účinná. Princíp podvzorkovávania spočíva v rozdelení obrazu na štvorcové plochy danej veľkosti, u ktorých sú sledované lúče vysielané len na ich vrcholy. Pre ďalšie spracovanie plochy prebehne rozhodovací proces, ktorý podľa výsledkov sledovania lúčov na jej vrcholoch rozhodne, či bude štvorec ďalej delený na menšie plochy alebo bude rovno vyfarbený farbou jej vrcholov. Po dosiahnutí maximálneho zanorenia tejto metódy sa plocha už ďalej nedelí a farba jej obsahu je získaná už len postupným vysielaním jednotlivých lúčov. Samotné podvzorkovanie dokáže urýchliť vykresľovanie až o niekoľko násobkov, ale jeho zlomový úspech spočíva najmä v kombinácii s bilineárnou interpoláciou, ktorá je síce náročná a citlivá na rozhodovací proces, ale prináša výsledné urýchlenie až o 10 a viac násobok pôvodnej rýchlosti.

4 Návrh

Po naštudovaní teórie a techník potrebných pre zvládnutie zadania, bolo nutné vypracovať návrh, na ktorý by nadviazal najobtiažnejší bod práce, a to vlastná implementácia *ray traceru*, ktorý by pracoval dostatočne rýchlo a efektívne zároveň so zachovaním vhodného stupňa vizuálnej kvality zobrazovaného obrazu. Keďže drvivá väčšina teórie bola po matematickej stránke netriviálna, očakával som najmä v počiatkoch implementácie značné problémy, preto som časti správneho návrhu vyčlenil dostatok času. Po skúsenostiach najmä z tímových projektov počas štúdia som vedel, že akékoľvek chyby v návrhu sa pri implementovaní vymstia niekoľkonásobne a stoja vývojára okrem strateného času aj psychickú pohodu pri samotnom programovaní a riziko znehodnotenia celej implementačnej časti a návratu k opätovnému spracovaniu správneho návrhu.

Tiež sa pri riešení tejto úlohy sa potvrdilo, že pri venovaní dostatočnej pozornosti plánovaniu, sa samotné programovanie uľahčí riešením dielčích problémov, ktorých problematika príliš neovlivňuje okolité celky a ideálne je izolovaná uprostred jednej triedy alebo metódy. Za optimálnych podmienok, pri detailne spracovanom návrhu by potom programovanie mohla byť už len značne mechanická činnosť. Bohužiaľ v tomto prípade implementáciu komplikovala závislosť každého celku na konkrétnom matematickom modeli. Vďaka nezanedbanému návrhu som, ale vzniknuté problémy mohol riešiť izolovane, čím sa minimalizovali ich dôsledky a rozsah z toho vyplývajúcich škôd.

4.1 Analýza

Pre naplnenie zámeru, pripraviť kvalitný plán štruktúry výsledného programu, som musel začať analýzou jednotlivých problémov spojených so zadaním. Mal som implementovať grafické intro, ktoré by pracovalo na princípe metódy sledovania lúčov a bolo obmedzené celkovou veľkosťou do 64kB.

Ako základné smery prekážok, ktoré bolo treba prekonať som videl :

- Implementácia *ray traceru*.
- Požadované zrýchlenie a jeho optimalizácia aby umožňoval prácu v reálnom čase.
- Splnenie pamäťového obmedzenia.
- Vytvorenie vizuálne príťažlivého intra pri dodržaní všetkých obmedzení, ktoré pracovali proti.

V tejto fáze vývoja som mal značné obavy zo zadaných požiadavkou, predovšetkým z nedosiahnutia dostatočne rýchleho *ray traceru* a z presiahnutia stanoveného pamäťového limitu. Preto som už pri návrhu myslel na aspekt rýchlosti a vhodnosť na implementáciu urýchľovacích metód. Návrh samotného jednoduchého *ray traceru* som mal uľahčení vďaka skutočnosti, že kvôli relatívnej netriviálnosti témy, je väčšina materiálov zaoberajúcich sa ňou doplnená niektorými vzorovými časťami kódu. Šlo napríklad o knihu *An Introduction To Ray Tracing [2]* alebo knihu *Realistic Ray Tracing [1]* , ktorá veľmi podareným spôsobom dokladala teóriu každej kapitoly kompletným kódom. Takto mohol čitateľ postupne naštudovať nie len teóriu ale zároveň si kapitolu po kapitole skladať svoj *ray tracer*. Bohužiaľ výsledný *ray tracer* bol už od začiatku navrhnutý so zameraním na maximálny vizuálny efekt, a preto pre moje účely nepoužiteľný. Podstatne použiteľnejší v tomto smere bol podobným štýlom pojatý návod dostupný na internete *Raytracing Topics & Techniques [6]*, ktorý bol síce tiež zameraný na príliš náročné vizuálne efekty a scény, ale jeho konštrukcia bola cez to o niečo vhodnejšia pre moje potreby. Preto som sa rozhodol pri plánovaní primárne vychádzať práve z tohto návodu a zároveň použiť niektoré časti kódu tu dostupné. Ako vhodný sa tiež ukázal stručný *ray tracer Aurelius [7]*, ktorého štruktúra bola pre študijné účely dobre čitateľná.

Vďaka tomu, že prvý bod bol analyzovaný s ohľadom na vhodnosť výsledného *ray traceru* na implementovanie akceleračných techník, bolo už v tejto fáze jasné, ktoré akceleračné techniky prichádzajú do úvahy. Vedel som, že pri daných obmedzeniach budem musieť pracovať s relatívne jednoduchou scénou o malom počte objektov, preto som predpokladal, že hlavná ťarcha urýchlenia padne na techniky znižujúce počet vysielaných lúčov a najmä na techniku adaptívneho podvzorkovania. Očakával som, že budem potrebovať viac ako 10 až 20 násobného urýchlenia, preto som vsadil na verziu podvzorkovania v kombinácii s bilineárnou interpoláciou.

Pre dodržanie pamäťového obmedzenia som musel pri plánovaní návrhu myslieť na minimalizovanie množstva kódu, použitie minimálneho počtu štandardných knižníc, a to predovšetkým objektovo orientovaných. Dôležitý bol tiež výber programovacieho jazyka a vývojového prostredia, pre jeho pamäťovú náročnosť a optimalizačné schopnosti. Ako vhodné sa tiež zdali programy na kompresiu binárnych súborov.

4.2 Vlastný návrh

Po dôkladnej analýze a vypracovaní zoznamu úloh a problémov, ktoré bude potrebné v priebehu implementácie riešiť, som konkretizoval jednotlivé funkčné celky výslednej aplikácie. Tieto som ďalej podľa požiadavkou rozdelil na menšie časti, s ktorými som ďalej pracoval a postupne ich od najnižšej úrovne spätne spájal do jedného celku.

- **Zobrazenie intra**

Do tohto celku patrilo okno aplikácie, ktoré málo slúžiť ako výsledný grafický výstup. Keďže intro je neinteraktívna aplikácia, nebolo potrebné implementovať rozsiahle grafické užívateľské prostredie a obsluha intra sa tak obmedzila len na jeho ukončenie.

- **Projekcia**

Sekcia Projekcia mala spravovať výstup vypočítaný ray-tracerom a zabezpečovať jeho prípadnú úpravu ako vkladanie textu.

- **Intro**

Táto časť mala naopak kontrolovať priebeh celého intra v skutočnom čase a zabezpečovať správu aktívnej scény.

- **Sledovanie lúčov**

Táto najrozsiahlejšia sekcia obsahovala jadro celého raytraceru, od správy vysielaných lúčov a, výpočtov ich prienikov cez riadenie sekundárnych lúčov, výpočtov osvetlenia a ďalších optických javov.

- **Kamera**

Kamera mala abstrahovať pohľad do scény a zabezpečovať smerovanie vysielaných lúčov.

- **Scéna**

Scéna mala uchovávať zoznam objektov, zdroj svetla a umožňovať abstrahovanie ich správy..

- **Objekt**
  - **Tvar**

    Sekcia Tvar mala obsahovať jednotlivé možné tvary objektov a poskytovať možnú prácu s objektmi, ktorá je jedinečná podľa konkrétneho tvaru, ako výpočet priesečníkov alebo normály.
  - **Materiál**

    Sekcia Material mala ukladať informácie o povrchových vlastnostiach objektu.
- **Zdroje svetla**

Sekcia spravujúcu nastavenia zdrojov svetiel.

- **Vektor**

Táto časť mala definovať základné matematické prvky nutné pre funkčnosť ray traceru a operácie s nimi. A to najmä vektor, ale tiež napríklad farbu alebo lúč.

5 Implementácia

Po dokončení analýzy a návrhu som sa rozhodol pre implementáciu v jazyku C++, najmä z dôvodu vhodnosti objektového programovania pre tento typ úlohy, tiež pre možnosti optimalizácie a v poslednom rade určite svoju úlohu zohrala osobná náklonnosť k tomuto jazyku.

V tejto kapitole rozoberiem vlastnú implementáciu zadania a budem sa postupne venovať riešeniam jednotlivých bodov, tak ako som ich špecifikoval v analýze. V jednotlivých častiach sa dotknem hlavných prvkov aplikácie, pričom v časti venovanej *ray traceru* budem postupovať oproti návrhu zdola nahor. Pre udržanie stručnosti sa pokúsim nezachádzať do prílišných detailov.

5.1 Vektor, lúč a farba

Základný prvok pre ďalšie výpočty a vôbec akúkoľvek prácu so scénou, ktorý bolo nutné definovať bol trojzložkový vektor. Tento je implementovaný triedou `vector3`, ktorej zložky sú reprezentované premennými `a`, `b`, `c`. Zároveň sú v tejto časti implementované typické vektorové operácie ako sčítanie, odčítanie vektorov, vektorový súčin, skalárny súčin, normalizácia a dĺžka vektoru a negácia.

Stavebným kameňom aplikácie je tiež farba užívajúca rovnako triedu `vector3` a lúče, ktoré sú implementované v triede `Ray` a určené normalizovaným vektorom smeru a počiatočným bodom.

5.2 Scéna a jej objekty

Scéna je reprezentovaná triedou `Scene`, ktorá spravuje množinu objektov v nej sa nachádzajúcich a implementovaných triedou `Object`. Zabezpečuje generovanie jednotlivých objektov počas prípravy scény na beh intra a to volaním metódy `InitScene` spolu s parametrom pre výber jedného z pripravených prednastavení. Ukazatele na takto pripravené objekty sú uložené v poli pravé aktívnych objektov `m_Objects`. Pretože tento *ray tracer* vznikal primárne pre mnou pripravované scény vhodné na zobrazenie intra v reálnom čase, mohol som prácu so scénou prispôsobiť vlastným požiadavkám. Preto scéna ukladá zdroje svetla oddelene od ostatných primitív do poľa `m_LightSrc` a tak znižuje počet počítaných priesečníkov počas behu intra. Tiež umožňuje uloženie primitív slúžiacich ako pozadie do poľa `m_BackgrObjects` oddelene od primárnych objektov scény, čím umožňuje eliminovanie týchto objektov pri sledovaní tieňového lúča. Zároveň nám to dovoľuje predpokladať, že vzdialenosť prienikov týchto objektov je väčšia ako

vzdialenosť od akéhokoľvek primárneho objektu, a tak umožňuje ich vyradenie z ďalších testov v prípade, že bolo zasiahnuté ľubovoľné z hlavných primitív.

Scéna na reprezentáciu objektov využíva triedy `Object`, ktorá poskytuje odkaz na materiál a tvar objektu definovaných triedami `Material` a `Shape`. Zatiaľ čo trieda `Material` slúži len ako úložisko údajov o parametroch povrchu, tak `Shape` podľa konkrétneho tvaru poskytuje operácie typické pre daný geometrický prvok ako výpočet prieniku so sledovaným lúčom alebo výpočet normály v mieste zásahu povrchu objektu. Zdroje svetla sú od triedy `Object` oddelené, a to triedou `LightSource`.

Trieda `Scene` tiež okrem iného umožňuje výber zrkadlovej osvetľovacej techniky akou budú prvky v nej osvetľované. Na výber má z rýchlejšej Phongovho alebo náročnejšieho ale opticky komplexnejšieho Cook-Torrancovho osvetľovacieho modelu. Tiež napríklad dovoľuje vybrať farbu pozadia, ktorá reprezentuje prostredie mimo zasiahnutých objektov alebo stupeň ambientného osvetlenia.

5.2.1 Material

Trieda `Material` abstrahuje povrchové vlastnosti jednotlivých primitív, a to tým že ukladá hodnoty nutné pre výpočet optických javov pri dopade na ich povrch. Jedná sa v prvom rade o farbu reprezentovanú typom `Color`, ktorý vychádza z triedy `vector3`. Ďalej musí instancia tejto triedy poskytovať stupeň reflexnosti povrchu a stupeň priehľadnosti pomocou metód `GetReflection` a `Get Refraction`. V prípade, že by disponoval nenulovou priehľadnosťou, poskytuje počas výpočtu lomu index lomu daného materiálu. A nakoniec pokiaľ zasiahnutý bod neleží v tieni, musí pri výpočte osvetlenia *ray traceru* poskytnúť difúzne a odrazové vlastnosti.

V prípade Cook-Torrancovho modelu, sú vlastnosti materiálov rozšírené o hrubosť respektíve hladkosť povrchu `m_Roughness` a stupeň Fresnelovho odrazu `m_FresnelR` v mieste zásahu. Pričom vhodnou kombináciou týchto dvoch vlastností je možné simulovať široké spektrum typov materiálov a vernejšie tak zobraziť kovové alebo plastické povrchy.

5.2.2 Tvar

Jednotlivé tvary primitív zastrešuje abstraktná trieda `Shape`, z ktorej vychádzajú triedy reprezentujúce jednoduché geometrické primitíva ako guľa `Sphere`, rovina `PlanePrim`, nekonečný a končený valec `InfinCylinder` a `FiniteCylinder` alebo nekonečný a konečný kužeľ `InfinCone` a `FiniteCone`. Implementované sú tiež trojuholníky triedou `Triangle`, ktoré umožňujú napríklad modelovanie mnohouholníkov a v intre boli použité na zobrazenie ohraničenej roviny. Každá trieda dediaca triedu `Shape` definuje okrem práce s pozíciou objektu v scéne a jeho rozmermi, tiež metódu výpočtu priesečníkov lúčov a metódu poskytujúcu normálu v bode zásahu.

Implementácia tvarov bola poznačená požiadavkou na dostatočnú rýchlosť *ray traceru*, preto dovoľovala implementovať iba základné geometrické primitíva umožňujúce dostatočne rýchly výpočet priesečníkov. Na druhej strane pri budovaní rozsiahlejšej scény je možné aj s jednoduchými tvarmi modelovať zložitejšie objekty.

5.3 Kamera

Kamera spracovaná triedou `Camera` generuje pre *ray tracer* lúče vysielané do scény metódou `GetA_Ray` a zabezpečuje tak prechod od dvoj-dimenzionálneho súradného systému okna aplikácie do troj-dimenzionálneho súradného systému scény. Poskytuje tiež možnosť zmeny nastavenia kamery, napríklad pomocou metód `SetA_View`, a tak dovoľuje implementovať pohyb kamery v scéne, ktorý je abstrahovaný od samotnej metódy na sledovanie lúčov. Smerovanie lúčov do scény je závislé na pozícii kamery a jej nasmerovaní. Metóda *ray traceru* `PXLRender` poskytuje pri volaní `GetA_Ray` informácie o práve spracovávanom pixeli, podľa ktorého dôjde k odchýleniu lúča od smeru pohľadu kamery v závislosti na definovaných rozmeroch plátna kamery a nastavenej orientácii tohto plátna. Vhodnou zmenou týchto nastavení v čase je možné následne využiť v intre k pohybu kamery.

5.4 Sledovanie lúčov

Sledovanie lúčov prebieha atomicky pre každý pixel zobrazovanej plochy a je započaté vygenerovaním príslušného lúča, ktorý nesie údaj o počiatočnom bode a normalizovaný smerový vektor pridelený kamerou. Tým je obsluha predaná metóde `Raytrace`, ktorá najskôr vyhodnotí stupeň spracovávaného lúča a následne prechádza k testovaniu prienikov daného lúča voči aktívnym objektom scény, kde je okrem zasiahnutého objektu sledovaná tiež vzdialenosť a bod zásahu v súradnom systéme. V prípade, že k prieniku nedošlo s nijakým objektom, je ďalšie sledovanie zastavené a počítanému pixelu je priradená farba odpovedajúca pozadiu nastaveného v scéne.

Po vyhodnotení najbližšieho miesta zásahu, je tento bod prieniku spracovávaný po stránke osvetlenia. Tu dochádza ako prvé k vyhodnoteniu tieňov vysielaním tieňových lúčov voči svetelným zdrojom, kde sa sleduje či nedôjde pred dosiahnutím svetla k prieniku s iným telesom. Preto sa opakuje proces výpočtov priesečníkov, tento-krát voči tieňovému lúču a s rozdielom, že pri prvom úspešnom prieniku už nepotrebujeme pokračovať vo výpočtoch a bod vyhodnotíme ako nachádzajúci sa v tieni voči konkrétnemu svetelnému zdroju. Hneď po spracovaní tieňov je použitá jedná z implementovaných osvetľovacích techník, kde na výber je z obvyklého Phongovho modelu alebo komplexnejšieho Cook-Torrancovho modelu. Pretože rozdielnosť obidvoch techník spočíva v rozdielnom prístupe k výpočtu zrkadlovej zložky svetla, je difúzna časť osvetlenia pre obidva modely totožná. V prípade prírastku svetla difúznym osvetlením ide o jednoduchú rovnicu závislú na

rozdielu medzi normálou primitíva v mieste zásahu a smerom dopadu svetelného lúča podľa rovnice $r_D(\vec{\mathbf{N}}\bullet\vec{\mathbf{L}})$, kde r_D je koeficient difúzneho odrazu materiálu a $\vec{\mathbf{N}},\vec{\mathbf{L}}$ je normála a smer dopadajúceho svetelného lúča. Výsledný svetelný prírastok je potom závislí na povrchových vlastnostiach konkrétneho telesa a je pripočítaný k výslednej farbe, ktorá tak v tejto fáze tvorí kombináciu farby materiálu a svetelných zdrojov. U telies s nulovou zrkadlovou zložkou alebo nulovou difúznou zložkou, je tento výpočet vynechaný. Tento proces sa postupne opakuje pre každý zdroj svetla oddelene a na konci po spracovaní všetkých pôvodov osvetlenia je k výsledku pripočítaná prípadná ambientná zložka.

V prípade, že je materiál telesa reflexný alebo priehľadný a nebola prekročená maximálna hĺbka rekurzie, je k farebnej zložke následne pripočítaný výsledok sledovania sekundárnych lúčov, ktoré počítajú farebný prínos lúčov odrazených od povrchu alebo lúčov prechádzajúcich skrz objekt. Miera tohto prírastku je úmerne závislá stupni reflexívnosti, priehľadnosti. Takto je rekurzívnym vysielaním a sledovaním sekundárnych lúčov zaručené fotorealistické zrkadlenie objektov. Vysielanie sekundárnych lúčov tak umožňuje tiež zobrazovanie priehľadných telies spolu s fotorealistickým lomením svetla v závislosti na indexu lomu objektu a okolia. Sekundárnymi lúčmi získaná farba je následne podľa stupňa reflexnosti, priehľadnosti započítaná do konečnej farby, ktorá sa skladá ako z farebných prírastkov získaných osvetľovacími technikami, tak farebných prírastok získaných práve sekundárnymi lúčmi, či už odrazenými alebo lomenými.

Po dokončení všetkých výpočtov je takto získanou konečnou farbou vyfarbený počítaný pixel na zobrazovanej ploche.

5.5 Akceleračné techniky

Fakt, že *ray tracer* bol od začiatku pripravovaný pre prácu s jednoduchými scénami s malým počtom primitívnych objektov a s relatívne rýchlym výpočtom priesečníkov, nebolo vhodné implementovať techniky ako obalové telesá a priestorové hierarchie. Kde napríklad použitie obalových telies by samo o sebe urýchlenie neprinieslo, najmä po použití adaptívneho podvzorkovania a pri vytvorených scénach by doba nutná na spracovanie sledovaného lúča naopak narástla. Tu som mohol vychádzať z výhody, že *ray tracer* bol vyvíjaný s ohľadom na dopredu navrhnuté intro, a tak som mohol zo znalosti rozloženia objektov voči kamere implementovať niektoré netradičné úsporné opatrenia. Jedno z nich bolo už čiastočne popísané v časti 5.2, ide o rozdelenie objektov scény do troch rozdielnych skupín, a to uložením ukazateľov do polí `m_LightSrc`, `m_Objects`, `m_BackgrObjects`, čím je možné eliminovať zbytočné testy na priesečníky so svetelnými zdrojmi. Separovanie objektov pozadí zase umožňuje ukončiť testy priesečníkov v prípade, že bol lúč zadržaný niektorým z primárnych telies. Tiež je možné telesá slúžiace ako pozadie vylúčiť z testov pri sledovaní tieňových lúčov, jednak kvôli ich polohe a jednak

preto, že sa ani nepredpokladá, že majú slúžiť ako svetelné zábrany. Samozrejme tento prístup prináša nevýhodu v nutnosti dodržať správne rozloženie scény, na druhej strane ide o nastavenie voliteľné a pri implementovaní scény nie je nutné ho použiť. Urýchlenie výslednej aplikácie bolo tiež dosiahnuté vhodným nastavením prekladača, projekt bol implementovaný vo vývojovom prostredí *Microsoft Visual Studio 2008* s nastavením optimalizácie */O2 (Maximize Speed)* a */Ot Favors fast code*. Nakoniec na rýchlosť nemali podstatný vliv len akceleračné techniky, ale tiež správny návrh a hlavne výsledná stavba *ray traceru*, ktorá je implementovaná s ohľadom na šetrnosť výpočetnej kapacity procesora a zároveň pamäťových požiadavkou spustiteľného súboru.

5.5.1 Adaptívne podvzorkovanie

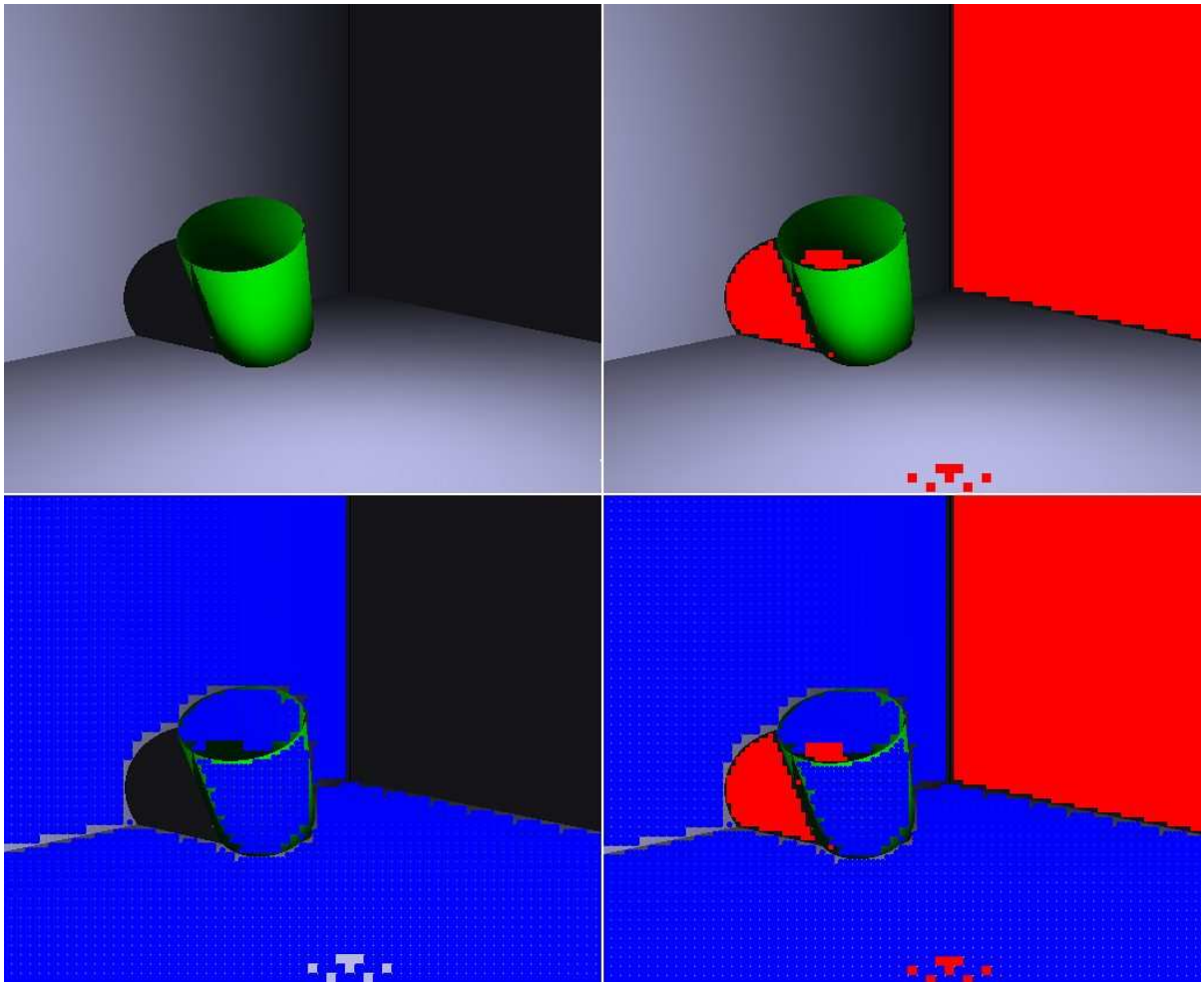
Adaptívne podvzorkovanie je riadené metódou `Render`, ktorá zobrazovaný obraz rozdelí na štvorce podľa žiadanej vzdialenosti medzi jednotlivými vzorkami. Následne pre každý takýto štvorec volá metódu `PXLSample`. Pre maximálne možné šetrenie výpočetného času, sa jednotlivé štvorce prelínajú, a preto táto metóda najskôr vyhodnotí, pre ktoré vrcholy štvorca nie je nutné opätovne získavať údaje. Následne sa postará o zavolanie metódy na sledovanie lúčov, ktorá zároveň uloží všetky potrebné údaje nutné k rozhodovacej činnosti o ďalšom postupe. Z týchto dát sú najskôr otestované body na farebnú totožnosť, kde je v prípade úspešnosti volaná metóda `PXLCopyColor`, ktorá len vyplní pixely medzi testovanými bodmi danou farbou. Pokiaľ prebehol test na farebnú totožnosť neúspešne, je stále možnosť že daný priestor bude môcť byť interpolovaný. Preto je volaná metóda InterpolTest, ktorá spracuje pre každý zo štyroch vrcholov, ich získané informácie. Následná interpolácia farby je preto podmienená tým, aby primárne lúče pretínali rovnaký objekt, kde v prípade, že došlo k odrazeniu musia sedieť zároveň objekty prienikov sekundárnych lúčov druhého stupňa. V prípade, že je zasiahnutý objekt priehľadný zaujímajú nás až sekundárne lúče tretieho stupňa. Pokiaľ sú splnené tieto podmienky, sú jednotlivé vrcholy postupne testované na výsledok získaný tieňovým lúčom voči všetkým svetelným zdrojom v scéne. Následne ten samý proces sa zopakuje pre osvetlenie miest zásahu sekundárnych lúčov, pokiaľ boli vyslané. V prípade, že všetky porovnania prebehli úspešne je daný štvorec vyhodnotený ako interpolovateľný, v opačnom prípade je v ktorejkoľvek časti test prerušený a vyhodnotený negatívne a k interpolácii nedôjde. Samotná interpolácia je volaná metódou `Interpolation`. Tá využíva bilineárnu interpoláciu, čo znamená že ako prvé dôjde k lineárnej interpolácii medzi dvoma a dvoma vrcholmi, čím získame dve vyfarbené krajné úsečky, medzi ktorými už môže prebehnúť opätovná lineárna interpolácia, tento-krát pre celú plochu štvorca.

Pokiaľ k interpolácii nedošlo je volaná metóda `PXLSubSample`, ktorá danú plochu rozdelí na štyri menšie a celý proces získavania vzorkou a ich spracovania sa pre každý z nich opakuje. V prípade neúspechu je delenie na podštvorce zastavené až dosiahnutím definovanej najmenšej plochy, ktorá je už braná ako ďalej nedeliteľná.

Tým že adaptívne podvzorkovanie pracuje v kombinácii s bilineárnou interpoláciou je táto akceleračná technika podstatne účinnejšia. Samotné vyfarbovanie na základe farebnej totožnosti je v scéne využívané skoro výhradne na hluché miesta bez objektov, kde je samotné sledovanie lúčov ukončene už po prvých testoch na priesečníky primárneho lúča, a preto nedochádza k tak výraznému ušetreniu výpočetného času ako u interpolovaných častí obrazu.

Ako vhodný kompromis medzi vizuálnou kvalitou obrazu a rýchlosťou sa ukázala veľkosť základného štvorca o stranách s desiatimi alebo ôsmimi pixelmi. Menšie plochy už vykazovali menšie urýchlenie. Naopak u rozmernejších plôch nie len, že dochádzalo k významným chybám v obraze, ale tiež od určitej veľkosti prinášali zároveň stratu na urýchlení.

Zmenšenie počtu sledovaných lúčov je vidieť na obrázku 5.1, kde červené plochy predstavujú pixely vyfarbované pri farebnej totožnosti vrcholov podvzorkovávaného štvorca, modré plochy predstavuj pixely vyfarbované bilineárnou interpoláciou. Počas testovania, pri použití prenosného počítača s procesorom Intel 1.73 GHz a jednoduchej scény, dosahovala táto akceleračná technika zhruba 14-násobného urýchlenia. Doba na vykreslenie kompletného obrazu o veľkosti 800x600 bez použitia adaptívneho podvzorkovania sa pohybovala okolo 1200 milisekúnd. Pri použití klasickej podvzorkovacej techniky došlo k urýchleniu na dobu okolo 450 milisekúnd a po zapojení možnosti interpolácie sa tento čas znížil až na dobu okolo 80 milisekúnd. Je nutné upozorniť, že tento pomer urýchlenia nemusí platiť pri všetkých zobrazeniach a je závislý na viacerých faktoroch. Bohužiaľ samotná interpolácia nesie aj negatívum generovaním určitých odchyliek a chýb, ktoré sú viditeľné najmä na pohybujúcich sa telesách, kde tak dochádza k zmene a preblikávaniu týchto fragmentov. Je možné tento problém buď vizuálne zmenšiť, zmenšením podvzorkovávaných plôch alebo úplne eliminovať sprísnením testov vhodnosti štvorca na interpolovanie. Bohužiaľ obidve riešenia vedú k čiastočnému, ale znateľnému spomaleniu, a pretože v prípade tohto *ray traceru* bolo pre mňa primárne zameranie na rýchlosť, rozhodol som sa túto chybu ignorovať.



Obrázok 5.1 – Demonštrácia implementovanej techniky adaptívneho podvzorkovania v kombinácii s bilineárnou interpoláciou. Červená predstavuje body vyfarbované na základe totožnosti farby, modrá predstavuje body interpolované.

5.6 Intro

Intro je prehrávané v okne o rozmeroch 800x600, ktoré bolo vytvorené pomocou funkcií knižnice WinAPI. Intro nie je pre užívateľa interaktívne a obsluhuje len užívateľské vstupy na jeho ukončenie. Keďže pôvodne bol význam intro chápaný tiež ako nekonečné demo, nie je časovo obmedzené. Striedajú sa tu preto dve scény v nekonečnej slučke. Dve scény preto, aby bolo možné demonštrovať viaceré zaujímavé techniky zvládané implementovaným *ray tracerom* pri použití jednoduchým scén s obmedzeným počtom objektov. Dve scény tiež umožňujú predstaviť dva rozdielne osvetľovacie princípy.

Samotný pohyb a zmeny či už objektov, svetiel alebo pohľadov je závislý na reálnom čase, a tak oddelený od aktuálneho výkonu počítača a počtu zobrazených snímkou za sekundu. Riešenie spočíva v počítadle skutočného času a jeho stopovaniu medzi jednotlivými snímkami. Preto nedochádza k typickému zrýchľovaniu respektíve spomaľovaniu pohybov. Tie sú realizované

prostredníctvom násobenia smerových vektorov odpovedajúcou konštantou menenou v čase. Keďže ide prakticky o dve spojené intrá, kde každé je samo o sebe v podstate nekonečné, sú konštanty v čase menené najčastejšie využitím trigonometrických funkcií.

Obrázok 5.2 – implementované intro, scéna 1

Prvá scéna trvá 32 sekúnd, je osvetlená využitím Phongovho osvetľovacieho modelu a obsahuje dva konečné ihlany, z toho spodný s otvoreným špicom. Medzi nimi sa pohybuje farebná guľa s dobrým stupňom reflexívnosti, ktorá demonštruje schopnosť *ray traceru* fotorealistického zrkadlenia. Táto uvádza do pohybu horný ihlan, ktorý potom samovoľne pomalým pohybom klesá. Na periférii týchto objektov krúžia nad sebou a v protismere dve priehľadné gule, modelujúce svojím vzhľadom sklený materiál, demonštrujúc tak opäť schopnosti *ray traceru* fotorealistického lomu svetla. Plocha na pozadí slúžiaca ako podklad má rovnako reflexívne vlastnosti a je definovaná ako jednoduchá rovina.

Obrázok 5.3 – implementované intro, scéna 2

Druhá scéna beží opäť po dobu 32 sekúnd a na svoje osvetlenie využíva Cook-Torrancov osvetľovací model. Celá scéna je nadefinovaná oproti prvej so značne svetlejším pozadím a farebnejšími telesami. Obsahuje tri čiastočne do seba vnorené reflexné gule zlatej farby, simulujúce kovový materiál. Tieto spolu tvoria akúsi prelievajúcu sa hmotu. Ich pohyb je na krajoch ohraničený dvoma rotujúcimi priehľadnými konečnými valcami a ako podklad v tomto prípade slúži štvoruholník zložený z dvoch samostatných trojuholníkov. Pre zatraktívnenie scény je ešte do intra zapojený pohyb kamery. Zaujímavým som tiež doznal pôvodne testovaciu verziu zobrazenia, kedy sú plochy vyfarbované technikou podvzorkovania, preto som po krátkych úvahách pridal tiež ich nepríliš dlhé štvor-sekundové zakomponovanie do intra.

Určite sa nejedná o intro na profesionálnej úrovni umeleckých výtvorov demoscény. No napriek tomu výsledné intro demonštruje vizuálne a technické možnosti vytvoreného *ray traceru*. A aj keď trpí jednoduchosťou scén, pokúsil som sa navrhnúť intro zaujímavé aj po stránke estetickej.

5.7 Veľkosť

Hranica, ktorú konečné intro nemohlo prekročiť, bola 64kB. Pre jej dodržanie bol projekt od začiatku navrhovaný a tvorený s ohľadom na jeho veľkosť. Tu sa ako kritické ukázalo eliminovanie používaných štandardných knižníc jazyka C++, ktoré sa hlavným pričinením podieľali na výslednej veľkosti spustiteľného súboru. Po ich nahradení štandardnými knižnicami jazyka C alebo úplnom odstránení, mala značný prínos na ďalšom zmenšení vhodná optimalizácia pri preklade zdrojového kódu. Čím sa podarilo získať intro o celkovej veľkosti 44544 bytov, ktoré už splňovalo limit. Ďalšie zmenšenie výsledku bolo možné dosiahnuť využitím programov na kompresiu veľkosti spustiteľných súborov. Tieto programy sú hojne využívané tvorcami demoscény, pracujú na princípe odoberania nepotrených dát z binárneho súboru, jeho následnou kompresiou spolu s priložením samorozbaľovacieho kódu. Po spustení sa tak aplikácia najskôr rozbalí do pamäte, odkiaľ je vzápätí spustená. Použitím *UPX - the Ultimate Packer for eXecutables*, ten dosiahol 45,98% ďalšieho zmenšenia a znížil tak výslednú veľkosť 20480 bytov.

6 Možné rozšírenia

Priestoru na ďalšie pokračovanie v projektu je určite veľa, tak ako v technických možnostiach sledovania lúčov, tak v samotnom grafickom intre. Okrem ďalšej optimalizácie kódu by bolo prínosné implementovanie nových akceleračných techník, ktoré sú buď menej známe alebo sa v budúcnosti objavia. Veľmi zaujímavé by bolo tiež využitie grafických kariet podporujúcich *ray tracing* po ich uvoľnení na trh, ale vôbec akákoľvek snaha o prenos výpočtov od procesoru, na ktorom v súčasnosti leží všetka ich ťarcha, ku grafickým adaptérom by mohol byť ten správny smer na ďalší vývoj. Vzhľadom na fakt, že každý bod obrazu je počítaný atomicky, bez vlivu na svoje okolie, je na mieste tiež úvaha nad sieťovým rozhraním a paralelizáciou sledovania medzi viacej počítačov. Otázka je nakoľko by také riešenie bolo vhodné pre grafické intro.

Pre zatraktívnenie samotného intra by malo značný prínos zakomponovanie hudby do dema, jednak by to znásobilo celkový estetický dojem a jednak je to súčasť, ktorá je u profesionálnej tvorby demoscény očakávaná. Ako ďalšiu možnosť vidím implementáciu zložitých objektov, napríklad použitie CSG modelov a celkové znáročnenie zobrazovaných scén. Ktoré by v takom prípade nútili k dosiahnutiu značného úspechu pri urýchľovaní aplikácie, preto tento smer vidím ako komplexnú výzvu.

7 Záver

Účelom tejto práce bolo vytvoriť grafické intro využívajúce techniku sledovania lúčov v reálnom čase a obmedzené veľkosťou na 64kB. Jednotlivé body zadania som splnil a vypracoval aplikáciu založenú na tejto technike, ktorá dosahuje dostatočnú rýchlosť pre prácu v reálnom čase a požadovaný počet zobrazených snímkou za jednotku času, tak aby bolo možné plynulé prekresľovanie intra na bežne dostupných počítačoch. Kde ako doporučené minimum sa pri testovaní ukázal 1,86GHz procesor, na ktorom intro o rozmeroch 800x600 pracovalo približne na 10 až 12 snímkoch za sekundu. Výkonnejší 2,8GHz procesor naproti tomu už pracoval s rýchlosťou cez 24 až k 32 zobrazeniam za sekundu. Aplikácia poskytuje tiež možnosť využitia optických javov typických pre metódu sledovania lúčov. Čoho sa snaží využiť výsledné intro, ktoré prezentuje technické možnosti implementácie. Pri jeho tvorbe bola, napriek značným obmedzeniam zobrazovanej scény, sledovaná tiež jeho estetická príťažlivosť. Posledný bod zadania, a to 64kB obmedzenie, vďaka vhodnému návrhu nepredstavoval výraznejšiu prekážku a bol tiež splnený.

Či už sledovanie lúčov alebo samotné intro, majú stále značné možnosti na ďalšie skvalitnenie či rozšírenie. Verím, že práca ako taká je vhodná pre ďalšie spracovanie, a že kód v nej implementovaný je pre takéto účely dostatočne prehľadný a otvorený.

Celý proces prípravy, návrhu a implementácie mal pre mňa veľký prínos najmä pre praktickú možnosť pracovať na projekte, ktorého zadanie bolo dostatočne široké aby ponechávalo podrobnosti implementácie na autorovej iniciatíve. Veľmi pozitívne tiež vidím, že som mal možnosť rozšíriť si svoje znalosti v oblasti počítačovej grafiky a previesť nadobudnuté teoretické znalosti do praxe.

Literatura

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

[2] Glassner S. A..: *An Introduction to Ray Tracing*. Morgan Kaufmann Publishers, San Francisco 2002.

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

[4] Mortenson M. E.: *Mathematics For Computer Graphics Applications*. 2nd edition, Industrial Press, New York 1999.

[5] Thalmann M. N., Thalmann D.: *Computer Animation*, 2nd revised edition. Springer-Verlag, Tokyo 1990.

[6] Bikker J., *Raytracing Topics & Techniques* [online], 2004, Dostupné na URL http://www.flipcode.com/archives/Raytracing_Topics_Techniques-Part_7_Kd-Trees_and_More_Speed.shtml

[7] Ing.Herout A., Ph.D.: *Materiály do predmetu Počítačová Grafika* na FIT v Brne.

[8] University of Cambridge: *Advanced Graphics and HCI* [online], Dostupné na URL http://www.cl.cam.ac.uk/teaching/1999/AGraphHCI/

[9] GD Wiki: *Cook-Torrance* [online], Dostupné na URL http://wiki.gamedev.net/index.php/D3DBook:(Lighting)_Cook-Torrance

[10] Demoscene portal: *demoscene.info* [online], Dostupné na URL http://www.demoscene.info/

[11] UPX: *the Ultimate Packer for eXecutables* [online], Dostupné na URL http://upx.sourceforge.net/

Zoznam príloh

Príloha 1. DVD so zdrojovými kódmi, spustiteľným súborom a plagátom.