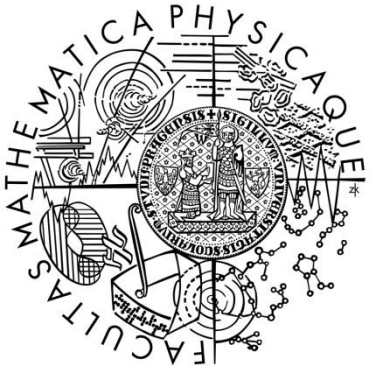
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Michal Hošala

Fotorealistické zobrazování

Kabinet software a výuky informatiky

Vedoucí bakalářské práce: RNDr. Josef Pelikán

Studijní program: Informatika

Studijní obor: Programování

Praha 2011

Na tomto mieste by som rád poďakoval všetkým ľuďom, ktorí mi boli psychickou oporou, najmä mojej priateľke a rodičom, počas obdobia, kedy som tvoril túto prácu. Vďaka patrí takisto vedúcemu mojej práce, RNDr. Josefovi Pelikánovi za jeho čas a rady pri konzultáciach aj mimo nich.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 25. 7. 2011

Název práce: Fotorealistické zobrazování

Autor: Michal Hošala

Katedra: Kabinet software a výuky informatiky

Vedúci bakalárskej práce: RNDr. Josef Pelikán

Abstrakt: Práca slúži ako teoretický základ pre implementovanie softvéru slúžiaceho k vykresľovaniu fotorealistických obrázkov. Aplikácia vyvinutá na základe popísaných princípov bude takisto súčasťou práce. Snaha o dosiahnutie fotorealizmu bude prebiehať na základe používania čo najpresnejších fyzikálnych, respektíve optických vlastností svetla a materiálov reálneho sveta a následného spracovania ich vzájomných interakcií. Práca sa pokúsi vierohodne spracovať a vykresliť vo výsledných obrazoch všetky možné druhy osvetlenia, vrátane osvetlenia nepriameho, ktoré popri priamom osvetlení tvorí v mnohých prípadoch takmer rovnako významnú zložku výsledných obrazov.

Kľúčové slová: ray tracing, photon mapping, KD strom

Title: Photorealistic image synthesis

Author: Michal Hošala

Department: Department of Software and Computer Science Education

Supervisor: RNDr. Josef Pelikán

Abstract: Goal of this thesis is to provide theoretical basis for implementation of the software to be used for rendering of the photorealistic images. Application based on described theoretical background is also part of this thesis. Photorealism is achieved by describing the light and materials in the scene, togehther with proper investigation of their interactions, by using as accurate physics/optics as possible, according to the real world properties. Thesis also aims on handling and following rendering of all kinds of illumination, including indirect illumination, which is often as important component of the final images as direct illumination.

Keywords: ray tracing, photon mapping, KD tree

Obsah

1. Úvod

Pod fotorealistickým vykresľovaním sa rozumie proces, pri ktorom sa v počítači vymodelované objekty zobrazia na monitor, alebo iné výstupné zaradenie. Najväčší dôraz pri tomto procese však nie je kladený na rýchlosť produkovania výstupu, ako je tomu bežne napríklad v počítačových hrách, ale na jeho kvalitu.

Za kvalitný výstupný obraz sa považuje ten, ktorý pôsobí hodnoverne, čo sa zväčša zabezpečuje dodržiavaním fyzikálnych zákonov o šírení svetla pri vykresľovaní, no nemusí to byť pravidlom. Je to preto, lebo ani spomínaný čas nie je zanedbateľným faktorom, a tak sa stáva, že výpočtový čas klesá, no táto úspora sa deje na úkor kvality. Väčšina techník, teda takmer všetky, totiž funguje na takej báze, že čím dlhšie tá ktorá aplikácia pobeží, tým kvalitnejší bude ponúkať výstupný obraz. Pri mnohých technikách bolo dokonca dokázané, že kvalita výsledku, dá sa povedať, s pribúdajúcim časom, ktorý má aplikácia k dispozícii konverguje k dokonalosti. No hneď po krátkom zamyslení sa, musí byť každému jasné, že čakať povedzme pár týždňov na vykreslenie jediného obrázku nie je žiadúce. Preto je jednou z prirodzených úloh fotorealistickej grafiky dneška hľadať kompromis práve medzi kvalitou výsledku a časom potrebným na beh aplikácie.

Osobitnú kapitolu pri hodnotení kvality fotorealistických obrázkov tvorí spracovanie nepriameho osvetlenia vo výsledku, na ktoré pri výpočtoch dlho nebol vôbec braný ohľad. Tento fenomén vzniká napríklad lomom svetla priehľadnými materiálmi, takzvaným pretekaním farby z jedného objektu na druhý a inými spôsobmi. Oba tieto efekty sú zobrazené na Obrázok 6, z ktorého bolo naviac vynechané priame osvetlenie, aby nepriame lepšie vyniklo.

1.1 Motivácia

Tvorba fotorealistických obrazov je hojne využívaná napríklad vo filmovom priemysle, či architektúre a jej vývoj neustále napreduje. Stačí si porovnať animované, alebo vedecko-fantastické filmy staré pár rokov s filmami dneška a tie staršie pôsobia miestami až komicky. Je to práve vďaka vtedajšej nedokonalosti počítačovej techniky, dostupnému softvéru, ale takisto kvôli tomu, že mnohé z techník na vykresľovanie v tých časoch ešte jednoducho neboli známe. Je to aj preto, lebo fotorealistická grafika sa veľmi dynamicky rozvíja a novinky z tejto oblasti sú nadnesene povedané dostupné takmer každý deň. Je preto náročné zachytávať všetky trendy fotorealistickej grafiky, no už teraz je bežné, že nielen človek nevenujúci sa grafike profesionálne, ale aj odborník, má v dnešných časoch veľký problém rozoznať niektoré obrázky vykreslené počítačom od tých, ktoré sú v skutočnosti fotografiami reálnych objektov reálneho sveta.

Silnou motiváciou je teda vysoká miera uplatnenia aplikácií určených na fotorealistické vykresľovanie v reálnom svete a takisto fakt, že táto vedná disciplína sa dá považovať za nikdy nekončiacu súťaž, v ktorej medzi sebou súťažia programátori celého sveta a snažia sa vytvoriť obrazy dokonalejšie a hodnovernejšie ako tí druhí.

1.2 Ciele

Táto práca má vytýčených hneď niekoľko cieľov:

- Na základe preštudovania dostupných zdrojov priblížiť problematiku fotorealistického vykresľovania, objasniť hlavné fázy, v ktorých toto vykresľovanie prebieha a priblížiť hlavné problémy, respektíve úlohy, ktoré musí aplikácia na to určená riešiť.
- Okrem presného počítania priameho osvetlenia spracovať aj problematiku osvetlenia nepriameho za využitia pokročilej techniky mapovania fotónov predstavenej v [1].
- Implementovať a popísať aplikáciu, v prostredí ktorej môže prebiehať ďalšie zdokonaľovanie produkovaných výstupných obrazov, založené napríklad na báze vylepšovania kvality, respektíve „fyzikálnosti“ materiálov, s ktorými vie aplikácia pracovať.
- Cieľom je takisto optimalizovať aplikáciu a popísať hlavné vylepšenia, a to aspoň do takej miery, aby bola po prípadnom ďalšom zdokonaľovaní vhodná aj na možné reálne využitie a nielen na študijné účely, pričom pod pojmom optimalizácia sa v tomto prípade rozumie optimalizácia aplikácie po algoritmickej stránke a nie po stránke využitia všetkej výpočtovej sily, respektíve prostriedkov, ktoré ponúkajú dnešné počítače.
- Na záver budú poskytnuté výsledky aplikácie, pričom budú konzultované ich jednotlivé atribúty, prednosti či nedostatky vo vzťahu k fotorealizmu, alebo inými slovami vo vzťahu k fyzike.

2. Globálne osvetľovanie

Táto kapitola bude venovaná úvodu do globálneho osvetľovania a niektorým konkrétnym technikám, ktoré sa používajú v počítačovej grafike na jeho výpočet, zreteľ však pritom bude samozrejme braný najmä na fotorealistické aspekty problematiky.

Mnoho ľudí si pri prvom kontakte s počítačovou grafikou ani nevie správne odpovedať na otázku, čo je najdôležitejším faktorom, ktorý určuje farbu objektov, na ktoré sa vo svete pozerá. Je to svetelné žiarenie, ktoré na daný objekt dopadá a je následne v upravenej forme odrazené od povrchu tohto objektu smerom ku pozorovateľovi, ktorý ho zrakovo vníma. Ako jednoduchý dôkaz môže byť použitý fakt, že ak na teleso žiadne svetelné žiarenie nedopadá, tak ho jednoducho nevidíme. Samozrejme to platí len v prípade, že objekt nie je sám o sebe svetelným zdrojom. V nasledujúcich podkapitolách si preto uvedieme, ako sa tento odraz svetla vypočítava a aké matematické vzorce založené na fyzikálnych respektíve optických vlastnostiach materiálov sa za tým skrývajú.

Vo zvyšku tejto práce sa často budú používať rôzne symboly, preto nasledujúca tabuľka popisuje ich význam. Hlbšie popísané budú niektoré z nich v ďalšej časti textu.

Symbol	Popis
x	Bod v priestore.
v	Jednotkový smer k pozorovateľovi.
l	Jednotkový smer do svetelného zdroja.
ω_p	Jednotkový smer do miesta, z ktorého fotón p dorazil na povrch.
ϕ_p	Svetelná energia, ktorú nesie fotón p.
n	Jednotkový normálový vektor v bode x.
L_l	Radiancia prichádzajúca zo smeru l.
L_v	Radiancia odrazená v smere vektoru v.
dv	Diferenciálny priestorový uhol v smere vektoru v.
f	BRDF.
f_d	Difúzna zložka BRDF.
f_s	Spekulárna zložka BRDF.
f_t	Transmisívna zložka BRDF.
Ω	Množina vektorov s počiatkom v strede pologule a koncom na jej plášti.
ρ_d	Difúzne albedo.
ρ_s	Spekulárne albedo.
ρ_t	Transmisívne albedo.

Tabuľka 1: Tabuľka často používaných symbolov spolu s vysvetlením.

2.1 Vzorce na výpočet globálneho osvetľovania

Problematika výpočtu šírenia svetla v scénach spadá pod súhrnný názov globálne osvetľovanie, pričom jedným z kľúčových pojmov tejto oblasti je radiancia, alebo L. Napríklad L_v je v skutočnosti vlastne to, čo vnímame vo výsledku ako farbu objektu.

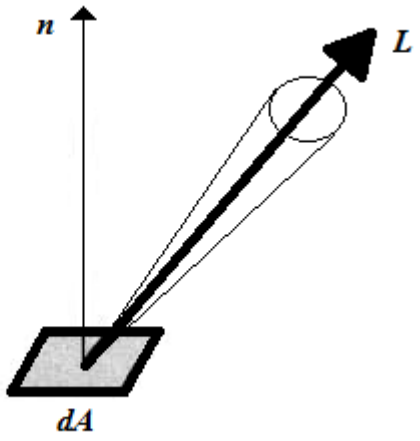
Na výpočet L_v bol vo viacerých publikáciách vrátane [1] navrhnutý nasledovný vzorec:

$$L_v(x,v) = \int_{\Omega} f(x,v,l)\, L_l(x,l)(ln)dl \qquad (2.1)$$

Slovami by sa dal popísať ako suma zo svetelného žiarenia, ktoré prichádza do bodu x zo všetkých smerov nad ním a je následne odrazené do smeru, kde je umiestnený pozorovateľ, teda v smere vektoru v. V nasledujúcich dvoch podkapitolách si v skratke priblížime dôležité pojmy radiancia a BRDF. Tieto, ako aj mnohé ostatné, vrátane celého úvodu do rádiometrie a fotometrie sú tak, ako vzorec (2.1) presne popísané, vrátane ich podrobného odvodenia v [1], odkiaľ boli čerpané aj mnohé nasledujúce informácie.

2.1.1 Radiancia

V Tabuľka 1 boli uvedené len špecifické radiancie L_v a L_l. Radiancia ako veličina však v oboch prípadoch vyjadruje to isté a takisto je uvedená v rovnakých jednotkách. Slovne sa jedná o množstvo svetelnej energie na jednotkový priestorový uhol na jednotkovú plochu. Situáciu dokresľuje Obrázok 1.



Obrázok 1: Radiancia L je definovaná ako žiarivý tok na jednotkový priestorový uhol, na jednotkovú plochu (zdroj: obrázok prevzatý z [1]).

Význam radiancie L_v sme si už vysvetlili v predchádzajúcej kapitole, no vysvetlenie pomocou práve uvedených tvrdení by znelo, že radiancia $L_v(x,v)$ je žiarivý tok opúšťajúci bod x v smere jednotkového priestorového uhlu okolo vektoru v. Radiancia $L_l(x,l)$ sa dá analogicky popísať, ako žiarivý tok prichádzajúci do bodu x zo smeru jednotkového priestorového uhlu okolo vektoru l.

2.1.2 BRDF

Táto skratka bola vytvorená z anglického názvu funkcie, ktorú predstavuje, a to „bidirectional reflectance distribution function“, čo v preklade znamená dvojsmerná distribučná funkcia odrazivosti. Po zvyšok tejto práce však bude spomínaná len ako BRDF. BRDF vyjadruje pre dvojicu vektorov v a l a bod v priestore x pomer medzi prichádzajúcou radiancou zo smeru l a radianciou, ktorá je potom odrazená do smeru v. Presný vzorec tohto pomeru je uvedený napríklad v [1]:

$$f(x, v, l) = \frac{dL_v(x, v)}{L_l(x, l)(ln)dl} \tag{2.2}$$

Jednou z veľmi dôležitých vlastností BRDF je jej reciprocita. Na jej dodržanie musí byť braný ohľad aj v implementácii svetelného modelu. Reciprocita znamená, že BRDF nadobúda rovnakú hodnotu, ak zameníme smery do svetelného zdroja a do pozorovateľa, exaktné vyjadrenie je [1]:

$$f(x, v, l) = f(x, l, v) \tag{2.3}$$

Ďalšou dôležitou vlastnosťou BRDF, ktorej nedodržanie môže v aplikáciách spôsobiť rôzne artefakty, je zákon zachovania energie, inými slovami, povrch telesa nemôže odraziť viac svetla, ako prijíma. Tento zákon vyjadruje [1]:

$$\int_{\Omega} f(x, v, l)\,(ln)dl \leq 1, \forall l \tag{2.4}$$

Ako už bolo uvedené v Tabuľka 1, je známych viacero druhov BRDF, ktoré nám po sčítaní dajú vo výsledku celkovú BRDF v danom bode. Tieto sú:

- Difúzna BRDF, ktorá určuje aká časť svetla je difúzne rozptýlená v bode dopadu. Difúzny rozptyl svetla vzniká vtedy, keď svetlo prenikne pod povrch a mnohonásobnými odrazmi tesne pod povrchom telesa je nakoniec spätne odrazené do pologule nad bodom dopadu.
- Spekulárna BRDF, ktorá určuje pomer svetla, ktoré je od povrchu odrazené v zrkadlovom smere.
- Transmisívna BRDF predstavuje zlomok svetla, ktoré prejde do telesa, no nenastane podpovrchové odrážanie, ako v prípade difúznej BRDF, lúč jednoducho pokračuje v zalomenom smere, ktorý sa získa pomocou Snellovho zákona[1].

Na presnú simuláciu difúzneho odrazu by bolo potrebné v implementácii použiť funkciu BSSRDF, ktorá presne simuluje podpovrchové odrážanie, nakoľko bez riadne implementovanej BSSRDF je nemožné rozlišovať medzi difúznou a transmisívnou BRDF. Preto v mnohých praktických aplikáciách materiál buď dokáže svetlo odrážať difúzne, alebo ho láme, nedokáže uskutočniť oba druhy interakcií svetla s jedným materiálom.

[1] Zákon lomu svetla, bližšie informácie dostupné na http://en.wikipedia.org/wiki/Snell's_law

2.1.3 Albedo

Výraz albedo ako taký sa používa na označenie miery odrazivosti povrchu daného telesa. V počítačovej grafike sa však zvykne používať s prívlastkom, pričom tento prívlastok určuje typ odrazu. Rôzne druhy albeda uvádzala už Tabuľka 1, pričom neboli bližšie vysvetlené, preto:

- Pre smer dopadu svetla l a normálu povrchu n vyjadruje difúzne albedo množstvo svetla, ktoré sa odrazí difúzne.
- Pre smer dopadu svetla l a normálu povrchu n vyjadruje spekulárne albedo množstvo svetla, ktoré sa odrazí spekulárne, teda zrkadlovo.
- Pre smer dopadu svetla l a normálu povrchu n vyjadruje transmisívne albedo množstvo svetla, ktoré sa na povrchu zalomí a prenikne do vnútra telesa.

Pre tieto albedá platí zákon zachovania energie, ktorý inými slovami hovorí, že povrch nemôže odraziť viac energie ako prijíma. Dá sa vyjadriť nasledujúcou rovnicou:

$$\rho_d + \rho_s + \rho_t \leq 1, \qquad \forall \rho_d, \rho_s, \rho_t \geq 0 \tag{2.5}$$

Zvláštnym typom albeda je ešte absorbčné albedo, ktoré vyjadruje, aká časť svetla bude po dopade na povrch telesa pohltená. Toto sa označuje ρ_a a vypočíta sa užitím vzorca (2.6):

$$\rho_a = 1 - \rho_d - \rho_s - \rho_t \tag{2.6}$$

2.2 Praktické riešenie globálneho osvetľovania

Pri pohľade na vzorec (2.1) je na prvý pohľad jasné, že vo svete počítačov v takejto forme nenájde uplatnenie. Je to kvôli tomu, že integrál v tomto prípade pracuje so spojitou funkciou, nakoľko svetlo môže prichádzať do bodu z nekonečne mnoho smerov v pologuli nad týmto bodom. Problém integrovania spojitých funkcií sa rieši takzvaným integrovaním Monte Carlo, pričom tento spôsob je dôsledne popísaný v mnohých materiáloch, ale napríklad aj v [1] a [2]. V skratke sa dá tento spôsob integrovania vysvetliť aj na príklade riešenia rovnice (2.1), pričom výpočet má niekoľko predpokladov a prebieha v niekoľkých krokoch, ktoré tu sú zachytené:

1. Predpokladom k výpočtu je to, že k intervalu, cez ktorý sa integruje, v prípade rovnice (2.1) to je Ω, je k dispozícia funkcia pravdepodobnostnej hustoty, takzvaná PDF[1]. V prípade rovnice (2.1) je teda potrebné poznať PDF pre výber smeru l, zo všetkých možných smerov nad bodom výpočtu.
2. Následne je zvolených niekoľko náhodných hodnôt z intervalu, cez ktorý integrujeme. Počet týchto vzoriek označme N, vzorky označme $l_1, \dots, l_N$.
3. V tejto fáze by sme pre každú takúto vzorku vypočítali hodnotu funkcie $g(l)$ pričom $g(l) = f(x, v, l)\, L_i(x, l)(ln)$. Hodnoty pre N rôznych vzoriek označme $g_1, g_2, \dots, g_N$.
4. Hodnotu integrálu následne aproximujeme ako:

[1] Viac o tejto funkcii na http://en.wikipedia.org/wiki/Probability_density_function

$$L_v(x,v) \approx \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \frac{\mathrm{g_i}}{pdf(l_i)} \tag{2.7}$$

5. Ako poznámku je dobré spomenúť, že čím vyššie je číslo N, tým presnejší bude výsledok, pričom pre N nekonečné výsledok konverguje k správnemu riešeniu.

Po tomto prevedení problému zo spojitého na diskrétny sa z neho stáva problém riešiteľný počítačom. V skutočnosti sa ale rovnica (2.1) ani jej aproximácia (2.7) nepoužívajú priamo na výpočet globálneho osvetlenia v scéne ako takého. Osvetlenie sa totiž zvykne rozdeľovať na viacero zložiek, ktorými sú osvetlenie priame a nepriame. Špeciálnu časť osvetlenia tvoria ešte kaustiky, ktoré by však logicky mohli spadať aj do skupiny nepriameho osvetlenia. Ich odhady sa však zvyknú počítať samostatne, preto aj v tejto práci im bude venovaná krátka osobitná sekcia.

2.2.1 Výpočet priameho osvetlenia

Priame osvetlenie tvorí vo väčšine prípadov najdôležitejšiu zložku vo výslednom obraze, takže na to, aby bolo čo možno najpresnejšie spočítané je kladený veľký dôraz. Na presné výpočty priameho osvetlenia slúži technika, ktorá sa nazýva tieňové lúče. Jej použitie za týmto účelom je niečo, na čom sa zhoduje zdrvujúca väčšina publikácií z prostredia fotorealistického zobrazovania, ako aj [1] a [2]. Na rozdiel od rovnice (2.1) výpočet priameho osvetlenia neprebieha tak, že sa do celej pologule nad bodom, v ktorom odhadujeme osvetlenie vyšlú náhodné lúče, ale lúče sú vysielané priamo do svetelných zdrojov, pričom sa zisťuje akým dielom ten ktorý svetelný zdroj prispieva ku výslednému osvetleniu. Tento prístup vyzerá na prvý pohľad logicky, nakoľko priame osvetlenie pochádza len zo svetelných zdrojov, a tak zo smerov kde sa žiaden svetelný zdroj nenachádza samozrejme nepríde žiadne svetlo priamo. Technika môže vyzerať jednoducho, no je potrebné brať do úvahy pár dôležitých predpokladov:

1. Pre každý svetelný zdroj sa dá na základe polohy bodu x a vektoru l smerujúceho do svetelného zdroja odhadnúť radiancia L_l, ktorá do bodu x prichádza. Táto radiancia môže byť pre svetelný zdroj zadaná ako konfiguračný parameter.
2. Svetelný zdroj dokáže produkovať rovnomerne rozmiestnené vzorky, body, na svojom povrchu. K týmto bodom je takisto potrebné poznať PDF toho ktorého bodu.
3. Je možné spočítať BRDF pre všetky materiály, ktoré sa v scéne nachádzajú.
4. Je možné zistiť viditeľnosť medzi dvoma bodmi v scéne, teda to, či sa medzi dvoma bodmi nachádza nejaká prekážka alebo nie.

Ak sú tieto predpoklady splnené tak už nič nebráni vo výpočte priameho osvetlenia na ktoromkoľvek mieste v scéne.

Výpočet by následne iteratívne prebehol pre všetky svetelné zdroje v scéne, pričom pre každý z týchto zdrojov by mal rovnaký charakter. Z tohto dôvodu bude na tomto mieste uvedený len výpočet osvetlenia od jedného svetelného zdroja. Ak by v scéne boli len bodové svetelné zdroje, tak by výpočet priameho osvetlenia od jedného takéhoto zdroja vyzeral nasledovne:

$$L_v(x,v) = f(x,l,v)\, L_l(x,l)(ln)vis(x, light_{origin}) \qquad (2.8)$$

Kde vysvetlenie jednotlivých symbolov sa nachádza v Tabuľka 1 a $vis(x,y)$ je funkcia, ktorá vráti 0 ak je medzi dvoma bodmi prekážka a 1 ak nie. Premenná $light_{origin}$ vyjadruje polohu vyšetrovaného svetelného zdroja v priestore. Priame osvetlenie od bodových svetelných zdrojov logicky vyúsťuje k tomu, že tiene v takto vykreslenom obrázku budú mať veľmi ostré okraje. Je to kvôli funkcii viditeľnosti, pretože pre ňu v prípade bodového svetelného zdroja neexistuje polo tieň. Bod sa nachádza buď v tieni alebo je osvetlený, viď nižšie.

Obrázok 2: Guľa osvetlená dvoma bodovými svetelnými zdrojmi vytvára na podlahe pod sebou ostro ohraničené tiene (zdroj: autor).

Rovnaký postup ako pre bodové svetelné zdroje, teda rovnica (2.8) by sa dal použiť aj pre povrchové svetelné zdroje, stačilo by na takom svetelnom zdroji pevne zvoliť jeden bod, voči ktorému by sme testovali viditeľnosť. Použitie rovnice (2.8) by však vyústilo v rovnaké výstupné obrázky ako použitie bodových svetelných zdrojov. Preto, ak je potrebné dosiahnuť efekt mäkkých tieňov, sa táto situácia rieši odlišne, a to tak, že:

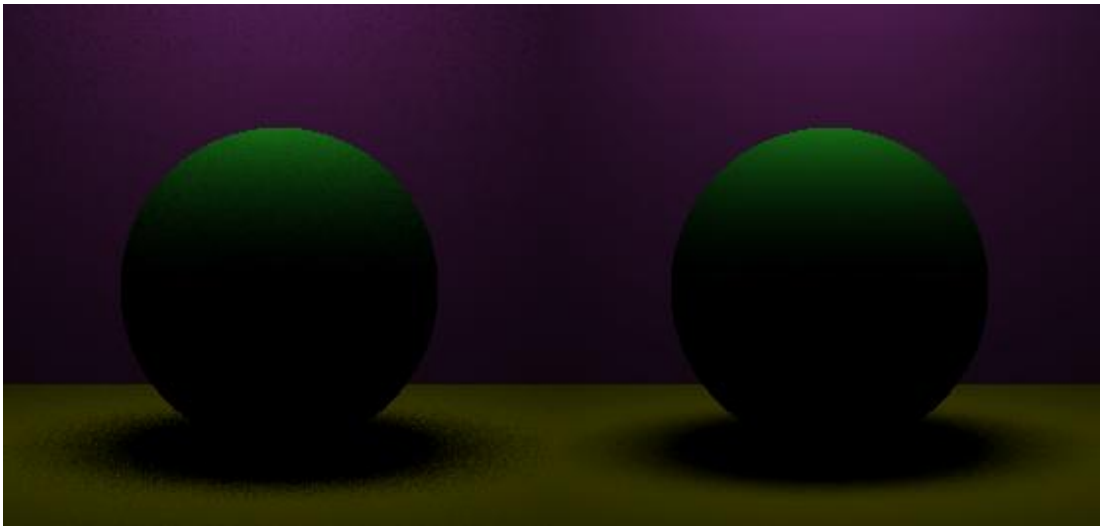
1. Na povrchovom svetelnom zdroji si zvolíme niekoľko náhodných bodov, povedzme $a_1, \dots, a_N$, pričom tým pádom budeme mať aj N rôznych $l_1, \dots, l_N$.
2. Pre týchto N rôznych hodnôt vypočítame N rôznych $L_{v,1}(x,v), \dots, L_{v,N}(x,v)$ užitím vzorca (2.8).
3. Výsledkom bude:

$$L_v(x,v) \approx \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \frac{L_{v,i}(x,v)}{pdf(a_i)} \qquad (2.9)$$

V rovnici (2.9) predstavuje funkcia $pdf(a_i)$ pravdepodobnostnú hustotu pre bod a_i, nakoľko voľba náhodného bodu z nejakej plochy je spojitá náhodná veličina a tak je táto funkcia pre každý bod v tomto prípade dobre definovaná. Rovnica (2.9) je

prvým dobrým príkladom použitia Monte Carlo systému integrovania popísaného v časti 2.2, ktorý sa používa v počítačovej grafike.

Prístup, ak by sa body pre výpočet rovnice (2.9) generovali na svetelnom zdroji úplne náhodne by mohol vyústiť k rôznym problémom. V počítačovej grafike sa za týmto účelom generuje napríklad 32, 64 alebo iný počet bodov na svetelnom zdroji, ktoré ak by boli rozmiestnené úplne náhodne, by mohli vyústiť do artefaktu vo výslednom obrázku známeho ako šum. Ten je viditeľný na Obrázok 3.

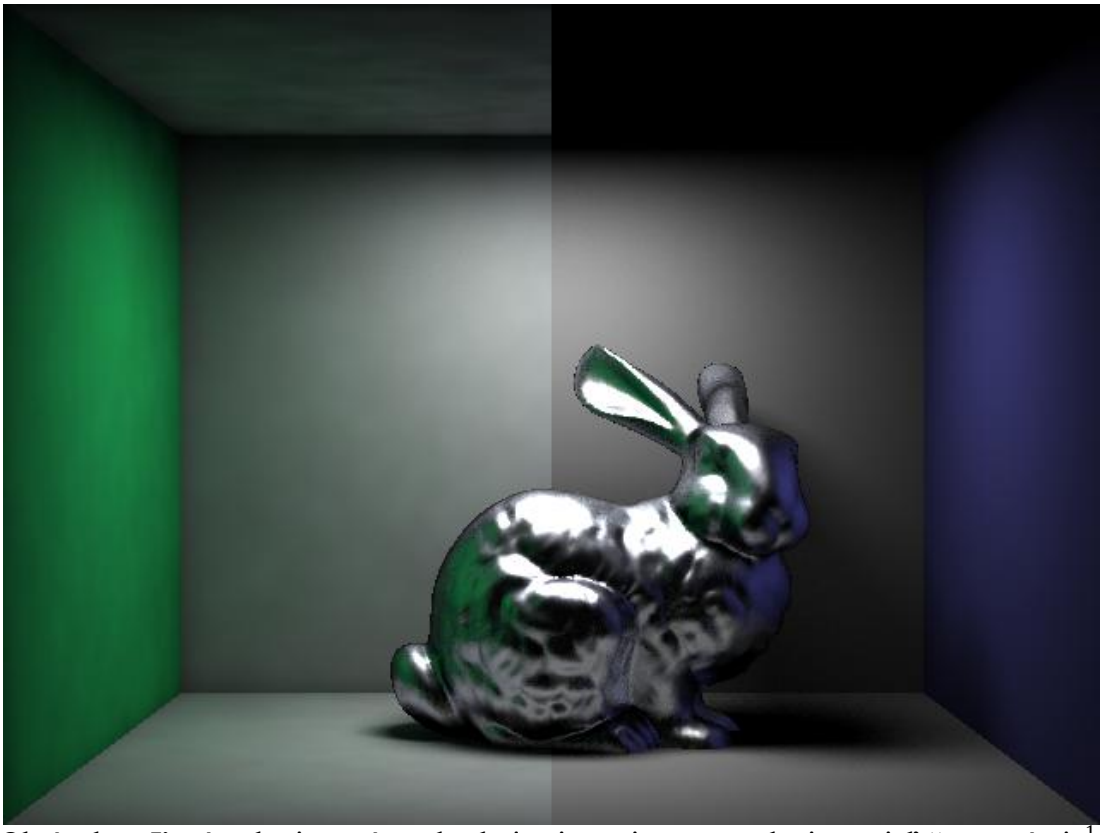


Obrázok 3: Mäkké tiene, v oboch prípadoch použitých 25 tieňových lúčov. Vpravo je šum redukovaný pomocou stratifikovaného vzorkovania (zdroj: autor).

K redukcii šumu existuje viacero prístupov. Jedným z nich je napríklad zvýšiť počet vzoriek, no táto technika nie je zďaleka taká úspešná ako napríklad stratifikované vzorkovanie a navyše konzumuje viac času na výpočet. Princíp stratifikovaného vzorkovania spočíva v tom, že plocha svetelného zdroja, napríklad obdĺžnik, sa rozdelí na toľko rovnakých neprekrývajúcich sa častí, koľko vzoriek je potrebných vygenerovať. Tieto časti sa potom vzorkujú samostatne, čo má za následok oveľa pravidelnejšie rozmiestnenie vzoriek, ako pri vzorkovaní celej plochy naraz a tým pádom aj kvalitnejší, teda menej zašumený obrázok.

2.2.2 Výpočet nepriameho osvetlenia

Nepriame osvetlenie na rozdiel od priameho nehrá vo výslednom obraze vykreslenom počítačom až takú podstatnú úlohu, no jeho prítomnosť či neprítomnosť môže veľmi významne ovplyvniť výsledok. Aj preto bolo vyvinutých niekoľko veľmi schopných techník, vďaka ktorým sa nepriame osvetlenie darí úspešne zobrazovať. Na začiatok je však vhodné spomenúť, čo to vlastne nepriame osvetlenie je. Jedná sa o typ osvetlenia, ktoré vzniká až po odraze priamych svetelných lúčov difúznymi povrchmi, teda osvetlenie, ktoré vytvárajú svetelné lúče, ktoré boli počas svojej cesty priestorom aspoň raz difúzne odrazené predtým, než dopadli na povrch, ktorý osvetľujú. Rozdiel medzi scénou, ktorá obsahuje okrem priameho aj nepriame osvetlenie je vyobrazený na Obrázok 4. Hlavným, respektíve najviditeľnejším rozdielom v tomto prípade je, že bez výpočtu nepriameho osvetlenia nie je vôbec osvetlený strop miestnosti, nakoľko svetelný zdroj sa nachádza tesne pod ním.



Obrázok 4: Ľavá polovica scény obsahuje aj nepriame osvetlenie, zatiaľ čo pravá nie[1] (zdroj: autor).

Na výpočet nepriameho osvetlenia navrhol Henrik Wann Jensen techniku, ktorú nazval photon mapping, alebo po preložení, mapovanie fotónov, pričom je podrobne rozobraná a popísaná v [1] vrátane čiastočného návrhu implementácie. Odhadovanie nepriameho osvetlenia použitím princípov zo spomenutej knihy by sa dalo v krátkosti zhrnúť do niekoľkých krokov:

1. V prvom kroku je potrebné na všetkých svetelných zdrojoch „vygenerovať“ isté množstvo fotónov, pričom každý fotón nesie časť svetelnej energie zdroja, z ktorého bol vystrelený. Súčet energií fotónov vystrelených z jedného svetelného zdroja nám dáva jeho celkovú energiu. To, koľko fotónov bude vystrelených z jedného svetelného zdroja záleží od jeho celkovej energie. Tým pádom je zo zdrojov, ktoré sú silnejšie, vystrelených viac fotónov.
2. V ďalšom kroku sú vygenerované fotóny „vystrelené“ do scény a zisťujú sa ich interakcie s telesami v scéne, pričom fotón, ktorý dopadne na difúzny povrch je zaznamenaný do dátovej štruktúry, ktorá sa nazýva fotónová mapa.
3. Po tom, ako skončí predchádzajúca fáza a fotóny sú úspešne uložené vo fotónovej mape je všetko pripravené na odhadovanie nepriameho osvetlenia.
4. Veľmi dôležitou poznámkou je to, že ak chceme z fotónovej mapy odhadovať nepriame osvetlenie, tak by sme do nej mali uložiť iba fotóny, ktoré toto osvetlenie vytvárajú. Konkrétne sú to fotóny, ktoré boli pred dopadom na

[1] Model stanfordského králika pochádza z http://graphics.stanford.edu/data/3Dscanrep/.

povrch, na ktorom budú zaznamenané, aspoň raz difúzne odrazené iným povrchom.

Generovanie fotónov

Fáza generovania fotónov je v podstate veľmi priamočiara a jedinú komplikáciu tvorí fakt, že má isté špecifiká pre každý svetelný zdroj, medzi ktoré patrí napríklad spôsob vzorkovania povrchu. Vo fáze generovania fotónov je totiž potrebné pre každý fotón získať jeho počiatočnú polohu a smer. Poloha sa získa presne spomínaným vzorkovaním povrchu, ktoré je napríklad pre obdĺžnikový svetelný zdroj triviálne. Smer je zvolený náhodne zo všetkých smerov na pologuli nad zvolenou pozíciou počiatku fotónu. Na tomto mieste je však nutné dbať na to, aby bolo použité vzorkovanie smeru, ktoré bude logicky zodpovedať výpočtu radiancie L_l, pričom napríklad ak tento výpočet nie je kosínovo vážený, tak nesmie byť kosínovo vážené ani vzorkovanie smeru generovaných fotónov. Kosínovo vážené vzorkovanie smeru znamená, že smery, ktoré zvierajú s normálou menší uhol budú zvolené s väčšou pravdepodobnosťou. Implementácia tohto vzorkovania vytvorená podľa návrhu z [2] je k nahliadnutiu nižšie.

```
PmVector sampleCosineWeightedUnitHemisphere()
{
    PmPoint sample = sampleUnitDisk();
    double x = sample.getX();
    double y = sample.getY();
    double zSquared = 1.0 - (x * x) - (y * y);

    return PmVector(x, y, sqrt(zSquared));
}
```

Obrázok 5: Kód v jazyku C++ predstavujúci kosínovo vážené vzorkovanie pologule nad normálou [0, 0, 1] (zdroj: autor).

Spracovávanie interakcií fotónov s telesami

Interakcie fotónov so scénou z predchádzajúceho zhrnutia si takisto zaslúžia viac pozornosti. Keď fotón dopadne na povrch telesa, tak môžu nastať štyri druhy interakcií, a to:

- Fotón môže byť difúzne odrazený.
- Fotón môže byť spekulárne odrazený.
- Fotón môže byť odrazený do vnútra telesa, teda zalomený.
- Fotón môže byť pohltený.

V skutočnosti by po každom dopade fotónu na povrch telesa malo nastať viacero z týchto situácií, pretože v skutočnom svete je v takom prípade napríklad časť energie pohltená a zvyšná časť sa povedzme rozdelí medzi difúzny a spekulárny odraz. Simulovať takéto interakcie by však bolo náročné na výpočet, nakoľko by sa takmer vždy po náraze fotónu na teleso stalo, že by bolo vytvorených viacero lúčov, ktoré by boli vrhnuté znova do scény. Počet fotónov čakajúcich na spracovanie by tak po každom ďalšom kroku exponenciálne rástol. Jensen preto v [1] navrhuje na zamedzenie tejto situácie použiť stochastickú techniku známu ako ruská ruleta. Ako ďalšie výhody, okrem urýchlenia, uvádza to, že zaručuje, že fotóny uložené vo fotónovej mape majú približne rovnakú energiu, čo je dôležité pre fázu odhadovania osvetlenia. K predpokladom na úspešné aplikovanie ruskej rulety patrí, že pre každý

materiál je známe akú časť svetla odráža spekulárne, akú difúzne a akú časť láme alebo absorbuje. Tieto štyri časti sa označujú ako $\rho_d, \rho_s, \rho_t, \rho_a$ a sú popísané už v kapitole 2.1.3. Na základe týchto hodnôt je potom možné ku interakcii fotónu s povrchom pristupovať štatisticky. Postup je nasledovný:

- Vygeneruje sa náhodné číslo $\xi,\ 0 \leq \xi \leq 1$.
- $\xi \leq \rho_d$ tak nastane iba difúzny odraz fotónu.
- $\rho_d < \xi \leq \rho_d + \rho_s$ tak nastane iba spekulárny odraz fotónu.
- $\rho_d + \rho_s < \xi \leq \rho_d + \rho_s + \rho_t$ tak nastane iba lom fotónu.
- $\rho_d + \rho_s + \rho_t < \xi$ tak nastane absorbovanie fotónu.

Znova, ako pri Monte Carlo integrácii, platí, že čím viac fotónov sa v scéne spracuje, respektíve do scény vystrelí, tým presnejšie výsledky budú touto technikou dosiahnuté. Konkrétne nové smery pre fotóny pri spracovaní interakcie s povrchom už závisia od materiálov použitých v scéne a od spôsobu, akým tieto materiály fotóny odrážajú, či lámu, čiže od ich implementácie.

Ukladanie fotónov

Ako už bolo spomenuté v krátkom zhrnutí tejto techniky, na ukladanie fotónov sa používa dátová štruktúra nazvaná fotónová mapa. Jedna z hlavných požiadaviek na ňu je, aby dokázala rýchlo vyhľadať fotóny uložené v okolí špecifikovaného bodu, v ktorom odhadujeme osvetlenie. Za týmto účelom je v [1] poskytnutá a plne popísaná dátová štruktúra, ktorá tieto požiadavky spĺňuje a pre nie natoľko pokročilé aplikácie je plne dostačujúca. Zber najbližších fotónov sa v nej dá ilustrovať ako postupné zväčšovanie gule so stredom v mieste, kde odhadujeme osvetlenie, až kým táto guľa neobsahuje chcený počet fotónov potrebných na odhad, alebo nepresiahne určitý maximálny polomer, pričom počet aj polomer určuje užívateľ. Tento spôsob zberu fotónov však môže v rohoch miestností vyústiť k nechcenému javu, kedy sa do odhadu započítavajú aj fotóny, ktoré neležia na ploche, na ktorej odhadujeme osvetlenie, ale napríklad na ploche na ňu kolmej a takisto sa do zoznamu najbližších fotónov dostanú aj fotóny nachádzajúce sa povedzme za stenou vo vedľajšej miestnosti vykresľovanej scény. Tento problém vzniká, pretože metóda na zber najbližších fotónov nepracuje natívne so scénou a tým pádom je jediným kritériom pri rozhodovaní, či fotón patrí do odhadu, jeho vzdialenosť od miesta, v ktorom vyšetrujeme osvetlenie.

Odhady osvetlenia z fotónových máp

Samotný spôsob odhadu osvetlenia z fotónovej mapy už závisí najmä od konkrétnej implementácie, pri ktorej je ale potrebné brať ohľad na nasledujúci vzorec prebratý z [1], ktorý slúži ako predloha na odhadovanie osvetlenia:

$$L_v(x, v) \approx \frac{1}{\Delta A} \sum_{p=1}^{N} f_d(x, \omega_p, v)\, \phi_p(x, \omega_p) \qquad (2.10)$$

Väčšina symbolov je vysvetlená v Tabuľka 1, no napríklad:

- Je dôležité si všimnúť použitú difúznu BRDF vo vzorci (2.10). Je to preto, lebo ako už bolo spomenuté, fotóny sa ukladajú len na difúznych povrchoch, čo ako tvrdí Jensen v [1] je plne logické, nakoľko spracovávanie zrkadlových

odrazov, či lomov zvládne najlepšie štandardný algoritmus ray tracingu, ktorému je venovaná kapitola 3.

- ΔA je závislé od telesa, ktoré bolo použité na zber fotónov. Predstavuje totiž obsah plochy, ktorá vznikne priemetom spomínaného telesa do roviny. V prípade, že je použitá guľa, ako je tomu v implementácii z [1], tak platí $\Delta A = \pi * r^2$, čo je inými slovami povrch kružnice s polomerom r, kde r predstavuje polomer gule, v ktorej boli nájdené fotóny určené na odhadovanie osvetlenia.
- N predstavuje počet fotónov, ktoré budú použité na odhadovanie osvetlenia, inými slovami fotóny, ktoré za týmto účelom vybrala fotónová mapa po zavolaní funkcie na nájdenie najbližších fotónov v okolí vyšetrovaného bodu.

V predchádzajúcej časti už bolo vysvetlené, že polomer gule, ktorá slúži na vyhľadávanie fotónov a maximálny počet fotónov v odhade určuje užívateľ. Toto však nie je až taká jednoduchá úloha a pre neskúseného užívateľa potrvá dlhšie, kým sa naučí tieto parametre správne voliť. K tejto problematike ale existuje niekoľko všeobecných rád:

- Ako tvrdí Jensen v [1] a iní, nepriame osvetlenie sa v scénach mení iba veľmi pozvoľna a neobsahuje nečakané, ostré farebné prechody, ako napríklad osvetlenie priame, kde sa striedajú oblasti v tieni a mimo neho. Preto, aby sa dosiahol tento hladký efekt nepriameho osvetlenia, sa zvykne zadávať väčší polomer gule, ktorá slúži na vyhľadávanie fotónov pre výpočet osvetlenia. Čím je tento polomer menší, tým viac artefaktov, vo forme fľakov, sa objavuje vo výslednom obraze. Tieto artefakty sú znázornené na Obrázok 6.
- Čím viac fotónov v odhade použijeme, tým kvalitnejší bude výstup.

Pre oba body však treba podotknúť, že so zvyšujúcim sa polomerom a počtom fotónov v odhade rastie samozrejme aj výpočtový čas. Preto je na tomto mieste vhodné ustanoviť primeraný kompromis medzi kvalitou a spotrebovaným výpočtovým časom. Tieto rozhodnutia sú však už plne na užívateľovi.

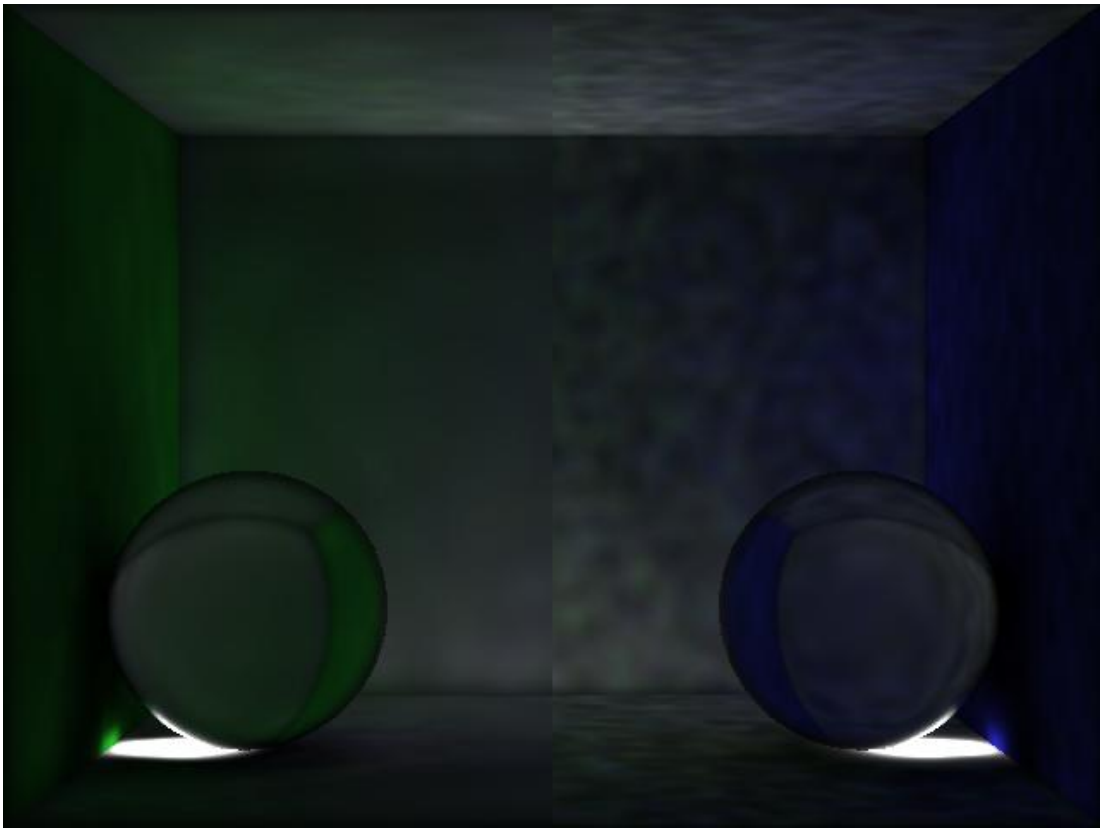
2.2.3 Výpočet kaustík

Kaustiky tvoria zvláštny druh, dá sa povedať, nepriameho osvetlenia. Nie však preto, že by súviseli s difúznymi odrazmi svetla, ale jednoducho preto, že nevznikajú pôsobením priamych svetelných lúčov, ale znova nepriamo, v tomto prípade po viacnásobných zrkadlových odrazoch, alebo lomoch svetelného lúča. Aby bol svetelný lúč označený ako kaustický, tak nesmie byť počas svojej cesty spracovaný inak, ako zrkadlovým odrazom alebo lomom, pričom takéto spracovanie muselo nastať aspoň raz, inak by sa jednalo o fotón priameho osvetlenia. Jav, ktorý pôsobením kaustických fotónov vzniká je v normálnom svete známy skôr pod hovorovým označením „prasiatko“.

Na tomto mieste potom môže vyvstať otázka, prečo sa teda kaustické fotóny neukladajú do fotónovej mapy spolu so zvyšnými nepriamymi fotónmi. Je to preto, ako už bolo spomenuté v predchádzajúcej podkapitole, lebo nepriame osvetlenie nepočíta s náhlymi, ostrými prechodmi medzi farbami, ale je prevažne hladké. Ako je však známe, kaustiky práve tieto ostré prechody medzi farbami vytvárajú, pretože

povrch, ktorý kaustické fotóny prijíma býva zväčša veľmi silno osvetlený v porovnaní s okolitým prostredím a tým pádom od neho aj výrazne odlíšený.

V spracovaní nepriameho a kaustického osvetlenia je však len veľmi málo rozdielov. Jedným z nich je, aby sa do fotónovej mapy na odhad kaustík ukladali len kaustické fotóny. Ďalším rozdielom je ten, že zatiaľ čo vyhľadávacia guľa pre nepriame osvetlenie má používať radšej väčšie polomery, tak pri kaustikách je to práve naopak, keďže tieto majú ostré hrany, ktoré by s použitím väčších polomerov nezostali zachované. Táto situácia je takisto znázornená na Obrázok 6, aj keď efekt rozmazaných okrajov kaustík nie je až natoľko viditeľný.



Obrázok 6: Vľavo nepriame osvetlenie odhadnuté s polomerom pre vyhľadávaciu guľu 0.6 a vpravo 0.3. V oboch prípadoch maximum fotónov v odhade 800. Vľavo kaustiky odhadnuté s polomerom pre vyhľadávaciu guľu 0.2 a vpravo 0.6. V oboch prípadoch maximum fotónov v odhade 750 (zdroj: autor).

3. Produkovanie výstupného obrazu – ray tracing

V predchádzajúcej kapitole bolo popísané, ako sa v scénach odhaduje osvetlenie, no to samozrejme ešte na produkovanie výstupného obrazu nestačí. Za týmto účelom sa vo fotorealistickej grafike používa takmer výlučne technika distribuovaného sledovania lúča, alebo jednoducho len technika sledovania lúča. V originálnom názve to je „ray tracing“ a tento pojem bude používaný na označenie tejto techniky vo zvyšku práce. Podstata spočíva v tom, že do scény je umiestnená priemetňa zložená z pixelov, na ktorú sa v závislosti na použitom spôsobe premietania premietne obsah scény. Pre stredové premietanie by táto časť prebiehala tak, že z ohniska premietania by sa cez každý pixel priemetne vyslal lúč do scény, ktorý by sa následne spracoval. Lúče vyslané z kamery sa nazývajú primárne lúče a ich ďalšie spracovanie, tak ako aj získavanie výslednej farby pre pixel bude popísané v ďalšej podkapitole.

3.1 Spracovávanie primárnych lúčov

Spracovávanie primárnych lúčov prebieha v niekoľkých krokoch:

1. Zistí sa, či nastal prienik lúča s telesom scény. Ak nie, všetky ostatné kroky sú preskočené a ako farba pixelu sa nastaví farba prostredia.
2. Nastal prienik. Vyhodnotíme priame osvetlenie v mieste dopadu, podľa rovnice (2.9) uvedenej v časti s priamym osvetlením, pričom za smer v je zvolený opačný smer, ako má práve vyšetrovaný lúč.
3. Vyhodnotíme nepriame osvetlenie a kaustiky v mieste dopadu, využívajúc rovnicu (2.10) uvedenú v sekcii o nepriamom osvetlení.
4. K aktuálnej farbe pixelu pripočítame hodnoty získané vyhodnotením osvetľovacích modelov v mieste dopadu.
5. Na základe optických vlastností materiálu zistíme, či zrkadlovo odráža svetlo. Ak áno, tak pomocou vzorca (3.1) spočítame zrkadlovo odrazený smer pre vektor v. Ďalej na základe vektorov v a n a konkrétneho materiálu zistíme, aká časť svetla sa zrkadlovo odráža a následne vyšleme do scény nový sekundárny lúč s takto získaným smerom a váhovými koeficientami.
6. Na základe optických vlastností materiálu zistíme, či láme svetlo. Ak áno, tak pomocou vzorca (3.2) spočítame lomený smer pre vektor v. Ďalej na základe vektorov v a n a konkrétneho materiálu zistíme, aká časť svetla, ktoré dopadne na povrch sa láme a následne vyšleme do scény nový sekundárny lúč s takto získaným smerom a váhovými koeficientami.
7. Sekundárne lúče spracovávame rovnako ako primárne, pričom ich príspevok ku farbe pixelu je najskôr prenásobený ich váhou, ktorú získali ochudobňovaním po odrazoch od určitých materiálov.

Hĺbka rekurzie pre spracovanie odrazených a lomených lúčov sa v zdrvujúcej väčšine aplikácií zvykne afyzikálne orezať, povedzme ak by počet odrazov a lomov prekročil číslo 3, alebo ak by váha priradená lúču bola nižšia ako 5%. Zabraňuje sa tak príliš dlhému výpočtu, pretože napríklad v zrkadlovej miestnosti by sa lúče vytvárali donekonečna a aplikácia by zostala v nekonečnom cykle.

Vzorce na výpočet odrazeného a lomeného smeru l, spomenuté v [1], ale aj takmer v každej dostupnej literatúre venujúcej sa aspoň okrajovo ray tracingu:

$$l = 2(vn)n - v \tag{3.1}$$

$$l = -\frac{n_{in}}{n_{out}}(v - (vn)n) - \left(\sqrt{1 - \left(\frac{n_{in}}{n_{out}}\right)^2 (1 - (vn)^2)}\right)n \tag{3.2}$$

Vo vzorci (3.2) predstavujú premenné n_{in} a n_{out} index lomu prostredia, z ktorého lúč prichádza, a s ktorým lúč koliduje. Inými slovami, n_{out} je index lomu materiálu, do ktorého lúč narazil a n_{in} je index lomu prostredia. Problém s takýmto riešením nastáva, ak index lomu materiálu nie je jednozložková veličina. Pre materiály si ho môžeme udržiavať napríklad v troch farebných kanáloch RGB, inými slovami červenom zelenom a modrom. Ak by sme pre každú túto zložku indexu lomu lámali vyšetrovaný lúč osobitne, tak by vo výslednom obrázku nastali veľmi nepekné artefakty, nakoľko by sa jednalo o akúsi nepodarenú difrakciu svetla, pretože svetlo by sa nerozložilo do celého spektra, ale len do troch farebných zložiek, ktoré by boli od seba ostro ohraničené. Na tomto mieste sa preto dá využiť isté zjednodušenie, ktoré by vo výsledku nemalo pôsobiť rušivo. Na lom lúčov sa jednoducho bude používať priemerný index lomu spočítaný z troch zložiek skutočného indexu lomu. Toto zjednodušenie napríklad pre vodu vyústi do menšej odchýlky pre jednotlivé kanály, ako 0.4%, nakoľko trojzložkový index lomu by mal hodnotu (1.337, 1.333, 1.331) a priemerný tým pádom 1.333667[1].

3.2 Drsné materiály – distribuovaný ray tracing

Distribuovaný ray tracing je súhrnný názov pre vylepšenú metódu základného ray tracingu, kedy je pri odhadovaní osvetlenia v nejakom bode potrebné vyhodnotiť integrál, pričom sa to rieši spôsobom podobným výpočetu mäkkých tieňov v kapitole 2.2.1. Použije sa teda niekoľko vzoriek, lúčov, na základe ktorých sa integrál vyhodnotí pomocou metódy Monte Carlo.

Načrtnutá idea o tom, ako vytvárať počas vykresľovania sekundárne lúče v mieste dopadu neberie ohľad na jeden fakt, a to, že povrchy telies v reálnom svete neodrážajú a nelámu lúče len v jednom, ideálnom smere. Tento jav je ilustrovaný na Obrázok 7. Takže, ak je povrch telesa drsný, tak je túto drsnosť potrebné nejakým spôsobom preniesť do tvorby odrazených a lomených lúčov. Implementačné špecifiká drsných povrchov, respektíve materiálov, ktoré s parametrom drsnosti pracujú, budú podrobne rozobraté v ďalšej kapitole venovanej výhradne implementácii materiálov. V tejto podkapitole si však priblížime ošetrenie odrazov a lomov drsnými povrchmi z pohľadu spomínaného distribuovaného ray tracingu. Výsledky po zapracovaní drsnosti povrchov do implementácie sú demonštrované na Obrázok 8.

Riešenie vyššie načrtnutého problému prebieha tak, ako je spomenuté v prvom odseku tejto podkapitoly, teda použitím Monte Carlo integrovania. Postup bude

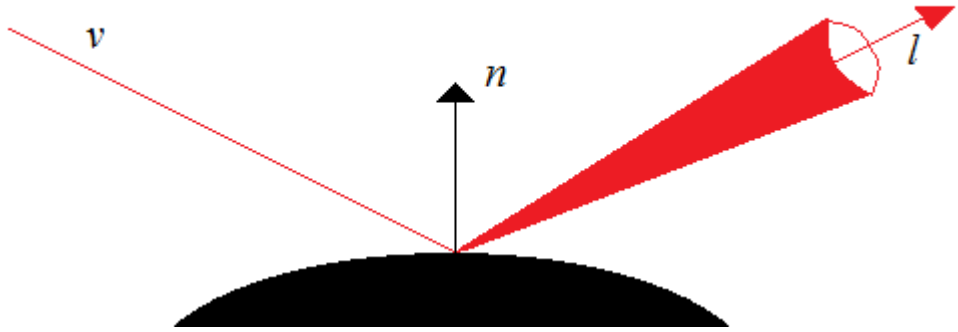
[1] Hodnoty pre index lomu získané zo stránky http://refractiveindex.info/

popísaný na vytváraní zrkadlového odrazu, no vytváranie lomu by bolo analogické. Ako interval, cez ktorý integrujeme sa v tomto prípade použije priestorový uhol znázornený na Obrázok 7, okolo ideálneho smeru odrazu l. Výpočet následne prebieha, ako už bolo spomenuté, v podobných krokoch ako výpočet mäkkých tieňov v kapitole 2.2.1:

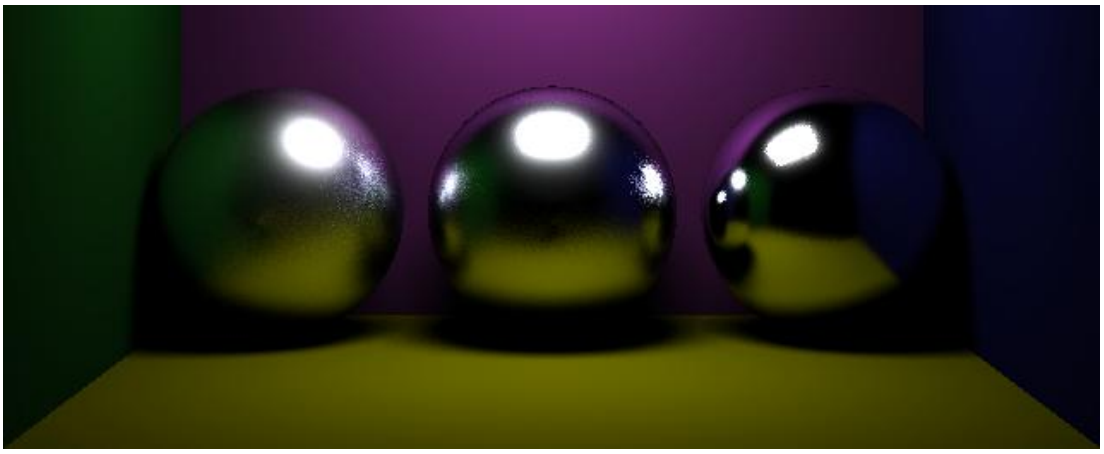
1. V rámci priestorového uhla je zvolených N rôznych smerov pre odrazený lúč, pričom tieto smery sú označené ako $l_1, \dots, l_N$.
2. Pre týchto N hodnôt je spočítaných N rôznych $L_{l_1}(x, l_1), \dots, L_{l_N}(x, l_N)$ ich vyslaním do scény a rekurzívnym vyhodnotením rovnakým spôsobom, ako sa to práve deje s primárnym lúčom v.
3. Finálny príspevok ku farbe pixelu od zrkadlovo odrazených lúčov bude predstavovať radiancia $L_{v,specular}(x, v)$ aproximovaná nasledujúcim vzorcom, prebratým z [3]:

$$L_{v,specular}(x, v) \approx \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \frac{L_{l_i}(x, l_i) f_s(x, v, l_i) |l_i n|}{pdf(l_i)} \qquad (3.3)$$

V tomto prípade predstavuje $pdf(l_i)$ funkciu pravdepodobnostnej hustoty, ktorá každému smeru z priestorového uhlu okolo l priradí istú hodnotu, zvyšné symboly sú vysvetlené v Tabuľka 1.



Obrázok 7: Lúč v, ktorý dopadá na povrch drsného telesa nie je odrazený v ideálnom smere l, ale "rozbíja" sa na nekonečné množstvo ďalších lúčov, ktoré sú odrazené do priestorového uhla okolo ideálneho smeru odrazu. Analogická situácia platí pre lom svetla (zdroj: autor).



Obrázok 8: Demonštrácia drsných materiálov v scéne. Všetky tri gule z hliníka, líšia sa len drsnosťou (zdroj: autor).

3.3 Urýchľovanie ray tracingu

Ako je z doterajšieho textu jasné, metódy vykresľovania založené na ray tracingu, poprípade distribuovanom ray tracingu, potrebujú vykonávať ako súčasť výpočtov veľmi často test na prienik telies v scéne s lúčmi, respektíve nájdenie najbližšieho bodu v scéne, ktorý daný lúč zasahuje. Jedná sa napríklad o primárne lúče, sekundárne, alebo tieňové, ktoré sa používajú pri výpočte priameho osvetlenia v scéne. Ak by sa v tomto prípade použil triviálny algoritmus na hľadanie najbližšieho prieniku, t.j. taký, ktorý by každý lúč testoval voči každému telesu, tak by technika ray tracingu pre väčší počet telies v scéne bola absolútne nepoužiteľná, výpočtový čas by bol jednoducho príliš dlhý. Z tohto dôvodu bolo na urýchlenie ray tracingu vymyslené množstvo techník, ktorých implementácia je dnes už podmienkou každého aspoň trocha seriózneho ray traceru.

Pred tým, než je v aplikácii implementovaná konkrétna urýchľovacia technika, je dobré najskôr čo najviac urýchliť počítanie prienikov s lúčmi pre jednotlivé druhy implementovaných telies. Na to neexistuje žiaden univerzálny spôsob, záleží to hlavne od konkrétneho druhu telesa. Spôsob ako počítať napríklad prienik trojuholníka a lúča však môže byť veľmi efektívne spočítaný využitím barycentrických súradníc[1].

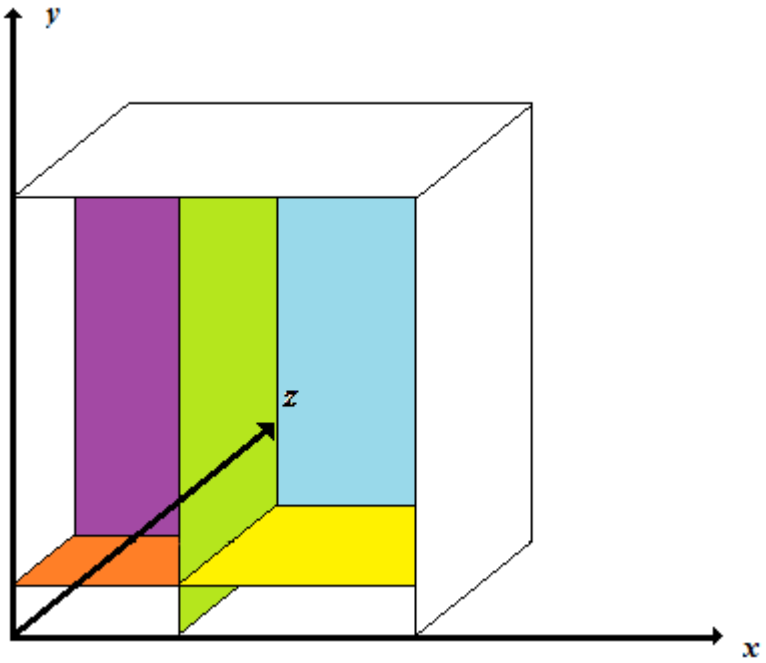
Po urýchlení jednotlivých prienikov je vhodný okamih na implementovanie nejakej pokročilejšej urýchľovacej techniky. Vhodnými kandidátmi sú BVH strom, alebo KD strom, pričom obe tieto techniky sú pomerne dôkladne rozobraté v [2]. Obe pracujú na takej báze, že priestor scény rozdelia na časti, následne sa zistí, ktorými časťami vyšetrovaný lúč prechádza a otestuje sa na prienik s telesami, ktoré v týchto častiach ležia. KD stromom sa veľmi dôkladne vo svojej dizertačnej práci venoval aj Havran v [4]. Na základe jeho štúdií a porovnaní bol preto KD strom vybraný ako vhodný kandidát na urýchľovaciu techniku, a preto je mu venovaná nasledujúca kapitola.

[1] Viac o tejto technike na http://www.devmaster.net/articles/raytracing_series/part7.php.

3.3.1 Urýchľovanie za použitia KD stromu

KD strom, ako už bolo spomenuté, je štruktúra, ktorá rozdeľuje priestor. Vo všeobecnosti dokáže rozdeľovať priestor v ľubovoľne mnoho rozmeroch, no v počítačovej grafike sa z logických dôvodov používa trojrozmerný variant. Rozdeľovanie priestoru potom prebieha vždy na dve polovice, pomocou deliacej roviny kolmej na nejakú z osí súradnicového systému. Po rozdelení priestoru na polovice sa tieto polovice ďalej rekurzívne znova rozdeľujú, až kým tento proces nie je istým kritériom ukončený. Delenie je znázornené na Obrázok 9. Algoritmus prebieha v niekoľkých základných krokoch, pričom každý z nich bude ešte osobitne popísaný:

1. Na začiatku sa spočíta kváder so stenami rovnobežnými so súradnicovými osami, ktorý slúži ako východiskový uzol ohraničujúci priestor, ktorý sa bude ďalej rozdeľovať, pričom tento uzol obsahuje všetky telesá, ktoré sa v scéne nachádzajú. Uzol je následne predaný procedúre, ktorá rozhoduje o jeho ďalšom spracovaní.
2. Procedúra, ktorá rozhoduje o ďalšom spracovaní má na starosť rozhodovať o tom, či uzol, ktorý dostala na vstupe ďalej rozdelí na dve časti pomocou deliacej roviny. Rozhodovanie i deliaca rovina sú spočítané na základe povrchovej heuristiky. Ak prebehlo delenie, tak sa rekurzívne znova zavolá táto procedúra na dva dcérske poduzly, pričom pre oba je najskôr spočítané, ktoré telesá do nich patria.
3. Po tom, ako nenastáva ďalšie delenie je KD strom postavený a pripravený slúžiť potrebám ray tracingu, pričom je potrebné implementovať akým spôsobom bude KD strom využívaný, inými slovami algoritmus na jeho prehľadávanie.



Obrázok 9: Delenie priestoru v KD strome pomocou rovín rovnobežných so súradnicovými osami (zdroj: autor).

Výpočet rozmerov východiskového kvádra – uzla

Predpokladom k tomuto výpočtu je to, aby sme pre každé teleso v scéne vedeli určiť jeho minimálne a maximálne súradnice v smeroch osí x, y a z. K tomuto účelu slúžia napríklad osovo orientované obalové telesá, ktoré sú na tento účel odporúčané aj v [2]. V angličtine sa nazývajú „axis aligned bounding box“, pričom miesto plného tvaru sa používa skratka AABB, ktorá bude používaná aj po zvyšok tejto práce.

AABB pre konkrétne teleso inými slovami predstavuje najmenší kváder so stenami rovnobežnými so súradnicovými osami, do ktorého sa to ktoré teleso celé zmestí. Logicky je teda AABB pre teleso určený šesticou čísel, ktoré sa však dajú zapísať aj vo formáte dvojice bodov, teda $bod_{min} = [x_{min}, y_{min}, z_{min}]$ a $bod_{max} = [x_{max}, y_{max}, z_{max}]$. Po tom, ako poznáme AABB pre každé teleso, je spočítanie rozmerov východiskového kvádra triviálne. Stačí už len nájsť minimálnu a maximálnu súradnicu v smere osí x, y a z pre AABB všetkých telies.

Rozdeľovanie uzlu – stavba KD stromu

Rozdeľovanie uzlu prebieha vo viacerých logicky oddelených častiach:

1. Zistí sa, pre ktorú zo súradnicových osí je aktuálny uzol najdlhší, teda na ktorej osi majú body bod_{min} a bod_{max}aktuálneho uzlu najväčšiu vzdialenosť.
2. Nájdu sa všetci kandidáti na danej osi, podľa ktorých by daný uzol mohol byť rozdelený. To prebieha tak, že sa prejdú AABB pre všetky telesá a zistia sa ich maximálne a minimálne súradnice na danej osi.
3. Pre všetkých kandidátov sa spočíta to, aké drahé bude rozdelenie aktuálneho uzlu podľa nich. Toto počítanie cien je založené na povrchovej heuristike, takzvanom OSAH, o ktorom píše Havran v [4]. Váha rozdelenia podľa konkrétneho kandidáta sa spočíta nasledujúcim vzorcom prebratým z [4]:

$$c_{split} = c_t + c_i n_l \frac{s_l}{s} + c_i n_r \frac{s_r}{s} \qquad (3.4)$$

Kde:

- c_t je cena traverzovania práve rozdeľovaného uzla. Ak hľadáme prienik lúča so scénou a lúč zasahuje tento uzol, tak táto cena vyjadruje zložitosť operácie zistenia, či lúč najprv zasahuje pravého alebo ľavého potomka. Havran v [4] navrhuje ako rozumnú cenu hodnotu 0.3.
- c_i je cena výpočtu prieniku lúču s akýmkoľvek telesom. Znova podľa Havranovho návrhu sa rozumnou hodnotou zdá byť 0.7.
- n_l a n_r predstavujú počet telies, ktoré sa po rozdelení uzlu budú nachádzať v ľavom poduzle/podpriestore, respektíve v pravom poduzle/podpriestore.
- s_l a s_r predstavujú povrch ľavého, respektíve pravého poduzlu.
- s predstavuje povrch súčasného uzlu.

Keďže uzly sú fakticky takisto reprezentovateľné a reprezentované ako AABB, teda dvojicou bodov, tak je počítanie povrchu triviálnou záležitosťou. Náročnejšie na výpočet je spočítať n_l a n_r, nakoľko v najhoršom prípade treba otestovať všetky telesá deleného uzlu voči prieniku s ľavým poduzlom

a takisto aj pravým poduzlom. Na urýchlenie výpočtu sa v tejto fáze často počítajú len prieniky AABB všetkých telies s poduzlami, namiesto prienikov telies s poduzlami.

4. Po tom, ako bola vypočítaná cena, c_{split}, pre každého deliaceho kandidáta, sa spomedzi kandidátov vyberie ten, ktorého cena je najnižšia. Následne sa spočíta cena toho, ak by sme daný uzol vôbec nerozdelili, na to slúži znova vzorec z [4]:

$$c_{nosplit} = c_i n \tag{3.4}$$

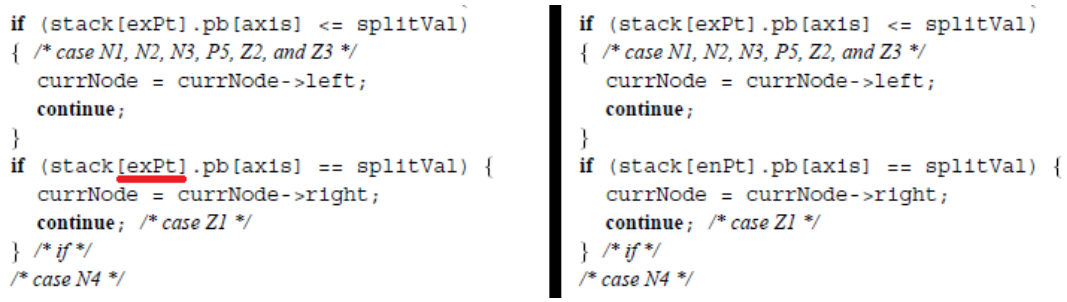
   Kde n je počet telies v aktuálnom uzle. Nasleduje rozhodnutie o tom, či vyšetrovaný uzol rozdelíme, pričom delenie nastáva len ak platí: $c_{nosplit} > c_{split}$.
5. Ak náhodou delenie nenastalo, tak sa algoritmus ešte vracia do kroku 1. a vyskúša celý postup znova s druhou najdlhšou osou, a ak by ani to nevyšlo, tak s treťou.
6. Ak nakoniec nastalo delenie uzlu, tak sa za určitých podmienok zavolá algoritmus rozdelenia uzlu rekurzívne aj na novovytvorené poduzly. Spomínanými podmienkami sú, že úroveň rekurzie neprekročila užívateľom stanovený limit, a že počet telies v poduzle neklesol pod úroveň takisto špecifikovanú užívateľom. Na hĺbku rekurzie je podľa [4] vhodné použiť napríklad hodnotu 24 a na minimálny počet telies v uzle použiť hodnotu 2.
7. Ak algoritmus skončí, napríklad ak je už nerentabilné ďalej uzly rozdeľovať, alebo bol prekročený jeden z limitov, tak je KD strom pripravený na použitie. Priestor je v tomto štádiu rozdelený do viacerých hierarchicky usporiadaných uzlov, pričom každý z uzlov vie, ktoré telesá obsahuje.

Prehľadávanie KD stromu

Podstatou prínosu použitia KD stromu spolu s ray tracingom je to, že lúč, ktorý testujeme na prienik so scénou, najprv otestujeme na prienik s KD stromom. Algoritmus na prehľadávanie KD stromu totiž dokáže nájsť postupnosť uzlov, ktoré vyšetrovaný lúč pretína, pričom tieto uzly sú nachádzané presne v poradí, v akom ich daný lúč navštevuje. Aplikácia sa tým pádom dostáva postupne aj k telesám, ktoré môže potenciálne pretínať vyšetrovaný lúč.

Mnoho konkrétnych prehľadávacích algoritmov je popísaných v [4], pričom ako najlepšia voľba sa javí rekurzívny TA^B_{rec}, ktorého implementácia v pseudokóde a popis je takisto prítomný v [4], implementácia v Appendixe C.

Autor tejto práce by však chcel dôrazne upozorniť na logickú chybu v danej implementácii, pričom táto je znázornená na Obrázok 10. Opravená verzia bola viacnásobne testovaná a poskytuje očakávané výsledky, preto si autor dovoľuje v rámci spomenutého obrázku poskytnúť túto opravenú verziu.



```
if (stack[exPt].pb[axis] <= splitVal)
{ /* case N1, N2, N3, P5, Z2, and Z3 */
  currNode = currNode->left;
  continue;
}
if (stack[exPt].pb[axis] == splitVal) {
  currNode = currNode->right;
  continue; /* case Z1 */
} /* if */
/* case N4 */
```

```
if (stack[exPt].pb[axis] <= splitVal)
{ /* case N1, N2, N3, P5, Z2, and Z3 */
  currNode = currNode->left;
  continue;
}
if (stack[enPt].pb[axis] == splitVal) {
  currNode = currNode->right;
  continue; /* case Z1 */
} /* if */
/* case N4 */
```

Obrázok 10: Chyba v Havranovom algoritme $\boldsymbol{TA}^{\boldsymbol{B}}_{\boldsymbol{rec}}$ z [4]. Chybný kus kódu je vľavo, pričom kód je aj bez širšieho kontextu na prvý pohľad chybný, keďže ak by bola podmienka pri druhom „if“ splnená, tak by bola splnená aj podmienka pre prvý „if“, v ktorom však je príkaz „continue“, preto sa k vyhodnoteniu tela druhého „if“ nie je možné nijak dostať (zdroj: obrázok prevzatý z [4]).

Spomínaný Havranov algoritmus TA^{B}_{rec} obsahuje okrem popísanej funkčnej chyby aj chybu, ktorá síce nemá dopad na jeho funkčnosť, no z hľadiska optimalizácie je potrebné ju brať do úvahy. Na jej pochopenie je potrebný aspoň krátky popis toho, ako pôvodný algoritmus pracuje:

1. Ak lúč, ktorý vyšetrujeme, pretína koreňový uzol KD stromu, tak je nájdený konkrétny poduzol stromu, ktorý je preťatý ako prvý.
2. Ak prvý preťatý poduzol neobsahuje žiadne telesá, tak sa postupuje k ďalším uzlom, ktoré sú po ňom preťaté.
3. V tomto kroku už je k dispozícii uzol, ktorý obsahuje nejaké telesá. Vyšetria sa preto prieniky lúča s týmito telesami a ak s niektorým z nich nastáva prienik, ktorý je bližšie, ako prípadné ostatné a navyše tento prienik leží vnútri vyšetrovaného uzlu, tak je tento prienik označený ako najbližší a vyšetrovanie končí. Ak ale prienik nenastal vnútri vyšetrovaného uhlu, tak je zahodený, aj keď je možné, že je skutočne najbližší, a vyšetrovanie prienikov pokračuje s ďalším uzlom, ktorý je v poradí.

Z bodu 3. je cítiť evidentné plytvanie časom, nakoľko môže nastať situácia, že nájdený prienik je ten správny, aj keď neleží vo vyšetrovanom uzle. Ak by teleso, na ktorom tento prienik nastal, ležalo aj v uzle, ktorý bude vyšetrovaný ako ďalší v poradí, tak bude musieť byť znova vyšetrený kompletný prienik lúču a tohto telesa. V [2] však v podobnej situácii, no s menej efektívnym a odlišným algoritmom na prehľadávanie KD stromu je toto plytvanie ošetrené, pričom použitá technika je aplikovateľná aj do Havranovho algoritmu TA^{B}_{rec}. Riešenie sa skladá z niekoľkých bodov:

- Každý lúč, ktorý je počas behu programu vytvorený má svoju jedinečnú identifikáciu ID.
- Každé teleso eviduje ID posledného lúča, s ktorým malo vyšetrovaný prienik, a tak sa algoritmus na prehľadávanie KD stromu môže opýtať každého telesa, ktoré sa nachádza vo vyšetrovanom uzle, či už náhodou nebolo testované na prienik s práve vyšetrovaným lúčom. ak áno, tak sa opätovné testovanie nevykonáva.
- V prehľadávacom algoritme KD stromu, ak je v bode 3. nájdený akýkoľvek prienik, tak je tento prienik evidovaný ako dočasne najbližší. Ak sa nájde iný prienik, ktorý je bližšie, tak sa informácie o prieniku prepíšu týmito novými informáciami.
- Ak je ukončené prehľadávanie nejakého uzlu a máme nájdený dočasný najbližší prienik, tak je tento prienik označený ako naozaj najbližší, ak

vzdialenosť do neho je pre vyšetrovaný lúč kratšia, ako vzdialenosť potrebná na opustenie práve vyšetrovaného uzlu.

Po krátkom pohľade na jednotlivé body a po ich pochopení si čitateľ môže jednoducho overiť fakt, že takýmto spôsobom sa pri vyšetrovaní prieniku lúčov so scénou zamedzí viacnásobnému počítaniu prienikov pre tú istú kombináciu teleso-lúč a že algoritmus bude takisto vracať správne výsledky.

4. Implementácia materiálov

Ako vyplýva aj z doterajšieho textu tejto práce, materiály a ich implementácia majú vo svete fotorealistickej grafiky veľmi dôležité postavenie, pričom kvalitné a fyzikálne verné spracovanie materiálov býva jedným zo zásadných faktorov pri hodnotení kvality dosiahnutého fotorealizmu výsledných obrazov v tej ktorej aplikácii.

Miesta pri výpočte fotorealistických obrazov, kde je potrebná implementácia materiálov, sú prítomné na každom kroku. Či už sa jedná o výpočet osvetlenia, priameho i nepriameho, výpočet zrkadlových odrazov a lomov, a tak podobne. Tieto miesta a spôsoby riešenia budú osobitne a podrobnejšie rozobraté v nasledujúcich podkapitolách, pričom na začiatok budú uvedené konštanty, ktoré sa využívajú pri popise materiálov.

Spracovanie materiálov z tejto práce vzniklo ako fúzia postupov navrhnutých v článkoch [3] a [5]. Obe tieto práce vzdialene vychádzajú z, v počítačovej grafike dobre známeho, mikroplôškového materiálu navrhnutého už v roku 1981 Cookom a Torrancom [6], a preto bolo možné ich využiť súčasne. Základným znakom materiálov založených na týchto publikáciách je to, že drsnosť povrchu nie je potrebné pre telesá modelovať, ale je vyjadrená štatisticky. Princípom tejto myšlienky je to, že pre každý bod na povrchu telesa vieme na základe štatistických údajov, ako „vyzerá“ jeho okolie. Každý bod sa totiž dá predstaviť ako nekonečne malá plocha, ktorá sa ďalej skladá zo spomínaných mikroplošiek, o ktorých vieme akým smerom sú orientované na základe štatistických údajov. Tieto údaje môžu byť vyjadrené napríklad distribučnou funkciou, ako je tomu aj v tejto práci.

Vzhľadom k používaným mikroplôškovým materiálom by mohol v tejto kapitole nastať zmätok v terminológii. Pre povrch materiálu totiž existujú dva typy normál, a to normála globálna a normála mikroplôšky. Tieto sú znázornené aj na Obrázok 12, pričom po zvyšok textu bude globálna normála označovaná n a normála mikroplôšky ako h.

4.1 Popis materiálov

Z pohľadu implementácie musí byť každý materiál reprezentovaný súborom konštánt, na základe ktorých sa s ním pracuje. Konštanty používané v navrhovanom riešení sú pre materiály zhrnuté v Tabuľka 2. Zvyšné často používané symboly v nasledujúcich podkapitolách boli zadefinované už v Tabuľka 1.

Symbol	Popis
n	Reálna zložka indexu lomu.
k	Imaginárna zložka indexu lomu.
d	Farba difúznej zložky materiálu.
m	Hrubosť, respektíve drsnosť povrchu materiálu.
α	Absorbancia materiálu, vypočítaná z k.

Tabuľka 2: Zoznam konštánt, ktorými sú popísané vlastnosti materiálov spolu s vysvetlením.

4.2 Ďalšie vlastnosti materiálov

Konštanty z Tabuľka 2 nie sú samé o sebe pre výpočty až také dôležité. Priamo sa totiž pracuje s hodnotami, ktoré sa vypočítavajú pomocou týchto konštánt. V nasledujúcich podkapitolách budú preto predstavené a vysvetlené najdôležitejšie vlastnosti materiálov z hľadiska výpočtov, ktoré budú vo výpočtoch v nasledujúcich kapitolách hojne využívané.

4.2.1 Fresnelov člen - F

Označuje sa F, pričom je vlastne funkciou. Táto funkcia berie ako argumenty globálnu normálu povrchu v určitom bode a smer, pod ktorým na tento povrch dopadol nejaký lúč vo forme jednotkových vektorov n a v. V závislosti na týchto dvoch vektoroch potom F vyjadruje, aká časť lúča bude od materiálu zrkadlovo odrazená naspäť do prostredia, z ktorého lúč prišiel. Obor hodnôt pre F je $[0, 1]$ a hodnotu je možné vypočítať napríklad na základe aproximácie poskytnutej v [7], ktorá bola vyvinutá špeciálne na to, aby bol Fresnelov člen presnejšie aproximovaný aj pre kovy, nakoľko tie majú vysokú imaginárnu zložku indexu lomu k. Na túto zložku však v pôvodnom vzorci na aproximovanie Fresnelovho členu, ktorý poskytol Schlick v [8] nebol vôbec braný ohľad, a preto padla voľba pre potreby tejto aplikácie na vylepšenú aproximáciu zo spomínaného článku [7]:

$$F(v,n) \approx \frac{(n_2 - n_1)^2 + 4n_2(1 - nv)^5 + k_2^2}{(n_2 + n_1)^2 + k_2^2} \tag{4.1}$$

kde n_2 a k_2 sú zložky indexu lomu pre materiál, na ktorý lúč dopadá a n_1 je zložka indexu lomu prostredia, z ktorého lúč prichádza.

4.2.2 Distribučná funkcia mikroplošiek - D

Vytváranie drsných povrchov materiálov použitím distribučnej funkcie pre mikroplôšky, ako sa spomína v úvode kapitoly 4 bolo použité už v [6], kde bolo navrhnuté za týmto účelom použiť Beckmannovu distribučnú funkciu predstavenú v [9] a popísanú rovnicou (4.2). Distribučná funkcia D ako taká má dve vstupné premenné, vektory h a n, pričom n definuje globálnu normálu povrchu a h je normálou mikroplôšky, pre ktorú je potrebné zistiť jej pravdepodobnostnú hustotu.

$$D(u,n) = \frac{\chi^+(un)e^{\frac{-tan\,(acos(un))^2}{m^2}}}{\pi m^2 (un)^4} \tag{4.2}$$

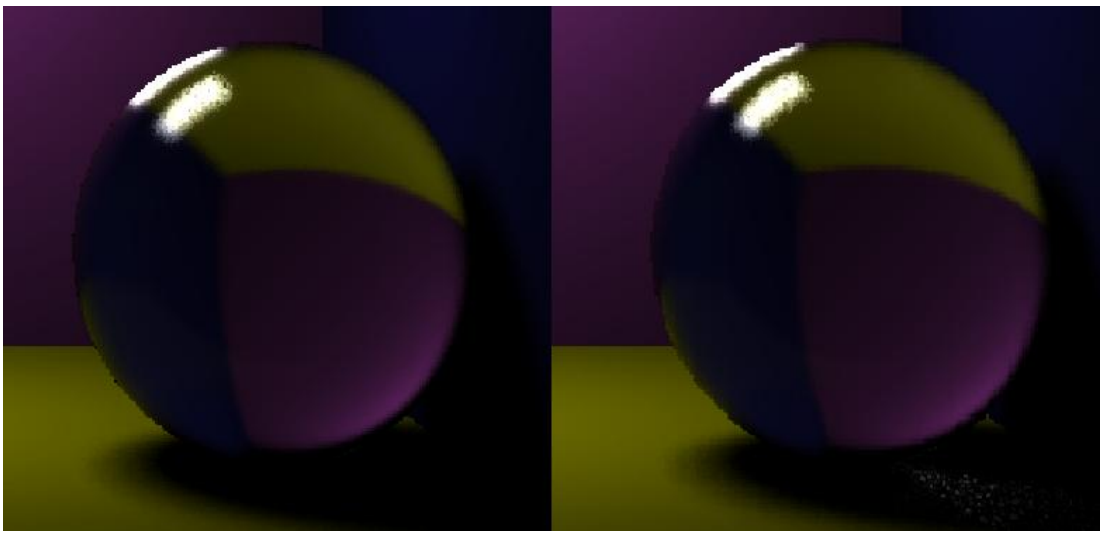
Beckmannovu distribúcia bola na tento účel použitá i v [3], no v to istom článku bola vyvinutá aj nová distribučná funkcia, ktorá podľa tvrdení v [3] viac vyhovuje požiadavkám implementácie priehľadných materiálov, a preto bola použitá aj v rámci tejto práce. Táto distribučná funkcia sa nazýva GGX a jej hodnota sa počíta ako:

$$D(u,n) = \frac{\chi^+(un)m^2}{\pi(un)^4(m^2 + tan(acos(un))^2)^2} \quad (4.3)$$

Jej použitie je v [3] odôvodnené tým, že pri vykresľovaní priehľadných materiálov je potrebné spracovať interakcie svetla na minimálne dvoch rozhraniach materiálov, a preto sa vyžaduje vyššia presnosť, ktorú GGX ponúka. Ako dôkaz tohto tvrdenia môže slúžiť, že aj počas práce na tomto projekte sa autor stretol s istými artefaktmi vo výsledných obrazoch, ak bola namiesto distribučnej funkcie GGX použitá Beckmannova distribučná funkcia. Tieto artefakty znázorňuje Obrázok 11.

Funkcia χ^+ vyskytujúca sa v oboch rovniciach (4.2) a (4.3) predstavuje funkciu pozitívnej charakteristiky daného čísla, pričom má obor hodnôt {0,1} a je definovaná ako:

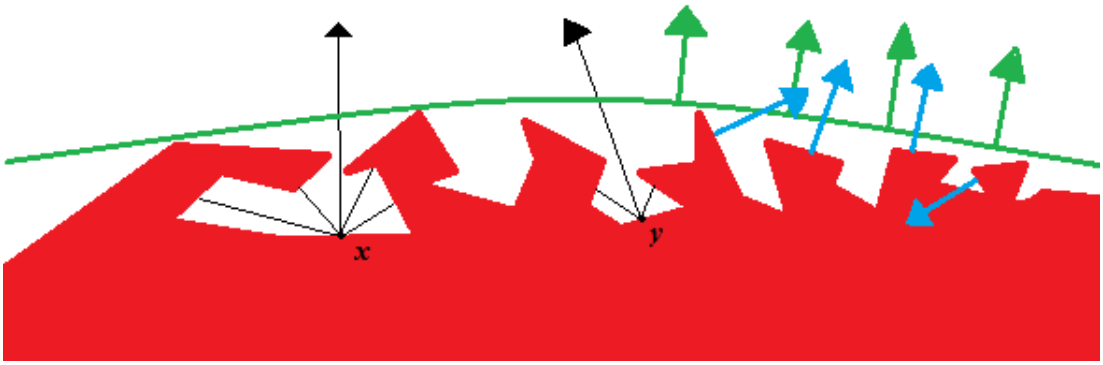
$$x \leq 0 \Rightarrow \chi^+(x) = 0,$$
$$x > 0 \Rightarrow \chi^+(x) = 1 \quad (4.4)$$



Obrázok 11 : Vľavo obrázok vykreslený použitím distribučnej funkcie GGX, vpravo pomocou Beckmannovej distribúcie. Na obrázku vpravo sú v tieni pod guľou viditeľné artefakty vo forme šumu. Nepriame osvetlenie bolo z výsledku naschvál vynechané, pre lepšiu viditeľnosť artefaktov (zdroj: autor).

4.2.3 Samozatieňujúci člen - G

Pre drsné materiály je potrebné simulovať jav, ktorý sa nazýva samozatieňovanie a je znázornený na obrázku nižšie.

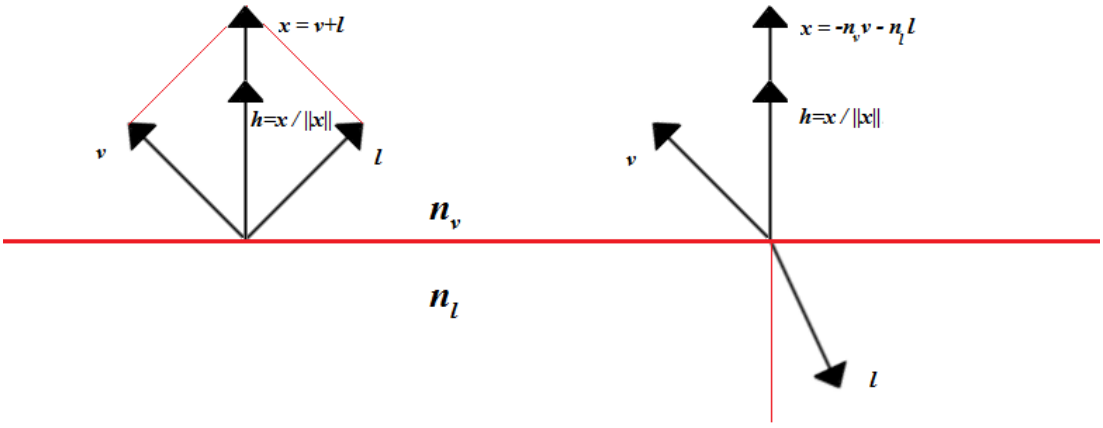


Obrázok 12: Zatieňovanie povrchu drsného telesa sebou samým. Body x a y sú vďaka mikro prekážkam tvoriacim drsný povrch čiastočne zatienené. Obrázok takisto znázorňuje normály mikroplošiek modrou farbou a normály povrchu z globálneho pohľadu zelenou farbou (zdroj: autor).

Pristupovať k tomuto problému modelovaním mikroprekážok tvoriacich drsné povrchy by bolo veľmi zdĺhavé a neefektívne. Preto bolo vyvinutých niekoľko samozatieňujúcich funkcií, ktoré pracujú na štatistickom princípe a samozatieňovanie povrchu určujú ako hodnotu z intervalu $[0, 1]$. V pôvodnom modeli popísanom v [6] sa na tento účel využívala nasledujúca funkcia:

$$G(v, l, h, n) = \min\left(1, \frac{2(nh)(nv)}{(hv)}, \frac{2(nh)(nl)}{(hv)}\right) \qquad (4.5)$$

Popis vektoru h je vysvetlený nižšie, pričom vzorce na jeho výpočet sú podľa vzoru [3] takisto uvedené priamo v obrázku.



Obrázok 13: Vľavo vektor h pre odraz lúča, vpravo pre lom. Vektor h v oboch prípadoch predstavuje normálový vektor, pre ktorý by vektory $\boldsymbol{v}$ a $\boldsymbol{l}$ boli v polohe ako na obrázku a platili by pre ne rovnice (3.1) a (3.2) pre výpočet odrazeného, respektíve lomeného lúča. n_v a n_l predstavujú reálnu zložku indexu lomu pre rozhranie daných dvoch materiálov, na ktorom nastáva odraz, respektíve lom lúča (zdroj: autor).

Funkcia v tvare (4.5) sa na výpočet samozatieňovania využívala aj v [5], no v [3] je zdôraznené, že táto funkcia sa na výpočet samozatieňovania nehodí, nakoľko jej

použitie môže vyústiť v prítomnosť artefaktov vo výslednom obraze, ktoré sa nevyskytujú v reálnom svete. Namiesto toho odporúčajú použiť Smithovu funkciu, ktorá úspešnejšie aproximuje samozatieňovanie. Smithova funkcia sa označuje G_1 , bola predstavená v [10] a aproximácia samozatieňovania je spočítaná pomocou nasledujúceho vzorca:

$$G(v,l,h,n) \approx G_1(v,h,n)G_1(l,h,n) \tag{4.6}$$

Ako je podrobne popísané v [3], táto Smithova funkcia sa pre rôzne použité distribučné funkcie D líši. Nakoľko za distribučnú funkciu bola v predchádzajúcej podkapitole zvolená funkcia GGX, tak aj nasledovný vzorec na výpočet Smithovej funkcie je určený pre použitie s GGX distribúciou (χ^+ viď rovnicu (4.4)):

$$G_1(v,h,n) = \frac{\chi^+\left(\frac{vh}{vn}\right)2}{1+\sqrt{1+\mathrm{m}^2 tan(\mathrm{acos(vn)})^2}} \tag{4.7}$$

4.3 Úloha materiálov pri výpočte priameho osvetlenia

Výpočet priameho osvetlenia popísaný v kapitole 2.2.1 pomocou vzorcov (2.8) a (2.9) obsahoval aj funkciu f, takzvanú BRDF, popísanú v kapitole 2.1.2, pričom táto BRDF je jednou z vlastností materiálu. V tejto podkapitole bude preto popísané, ako hodnotu tejto funkcie získať pre konkrétny materiál.

Tabuľka 1 uvádzala tri rôzne druhy BRDF a každý z nich je potrebné vedieť vypočítať osobitne. Je to preto, lebo BRDF sa nevyhodnocuje ako jednoduchý súčet f_d, f_s a f_t, ale rozlišujú sa dve situácie, a to:

- Svetelný model sa vyhodnocuje na povrchu telesa, pričom lúč od pozorovateľa do tohto telesa narazil. V tomto prípade bude BRDF tvorená súčtom f_d a f_s.
- Svetelný model sa vyhodnocuje na povrchu telesa, pričom lúč od pozorovateľa toto teleso opúšťa. V tomto prípade bude BRDF tvorená iba z f_t.

4.3.1 Spekulárna BRDF

Výpočet spekulárnej zložky BRDF na účely tejto práce pôvodne vychádzal z riešenia navrhnutého v [5]. Po zapracovaní priehľadných materiálov do projektu sa však toto riešenie stalo nepostačujúcim a celý výpočet bol prebratý z [3], nakoľko tento článok je venovaný práve priehľadným materiálom. Výpočet spekulárnej zložky BRDF je potom nasledovný:

$$f_s(x,v,l) = \frac{F(l,h)G(v,l,h,n)D(h,n)}{4|ln||vn|} \tag{4.8}$$

Pre vysvetlenie jednotlivých symbolov viď predchádzajúce kapitoly, Obrázok 13 a Tabuľka 1.

4.3.2 Difúzna BRDF

Postupy na výpočet rôznych foriem BRDF načrtnuté v [3] vôbec nepočítajú s difúznymi materiálmi a tým pádom ani s difúznou BRDF. Postupy však bolo možné spojiť so spracovaním materiálov a výpočtom difúznej BRDF navrhnutej v [5], pričom na jej výpočet sa používa nasledovný vzorec:

$$f_d(x,v,l) = d\frac{(1-\rho_s(v,n))(1-\rho_s(l,n))}{\pi(1-\rho_s^{ave})} \tag{4.9}$$

Kde ρ_s predstavuje spekulárne albedo popísané v časti 2.1.3 a ρ_s^{ave} predstavuje priemerné spekulárne albedo pre všetky možné uhly dopadu. K výpočtu týchto veličín pre ten ktorý materiál však ešte neboli uvedené všetky potrebné vzorce, a tak bude ich výpočet popísaný až v kapitole 4.5.1.

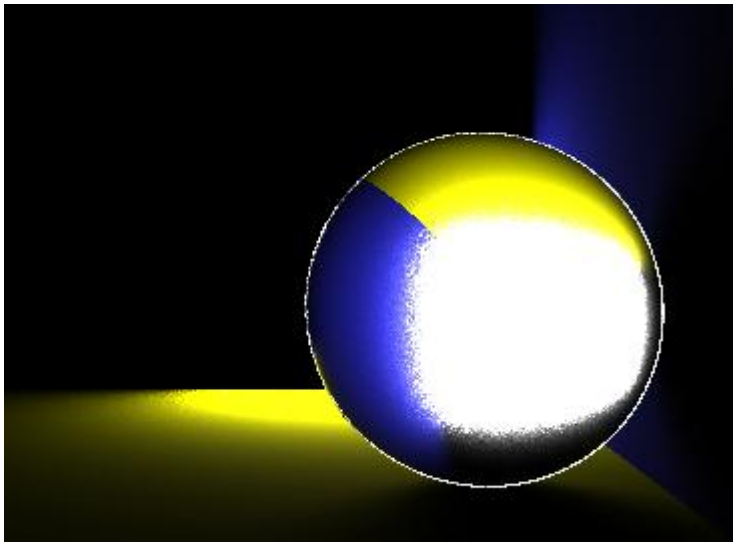
4.3.3 Transmisívna BRDF

Ako základ pre transmisívnu BRDF pre túto prácu poslúžil znova článok [3], pričom ako presný vzorec na výpočet bol použitý:

$$f_t(x,v,l) = \frac{|lh||vh|}{|ln||vn|}\frac{n_v^2(1-F(l,h))G(v,l,h,n)D(h,n)}{(n_l(lh)+n_v(vh))^2} \tag{4.10}$$

Pre vysvetlenie jednotlivých symbolov viď predchádzajúce kapitoly, Obrázok 13 a Tabuľka 1.

Počas tvorby tejto práce sa ale vyskytli s týmto výpočtom transmisívnej BRDF isté problémy, nakoľko jej použitie za určitých podmienok spôsobuje isté artefakty viditeľné aj na obrázku nižšie.



Obrázok 14: Scéna obsahuje sklenenú guľu, za ktorou je umiestnený plošný svetelný zdroj. Biela obruba okolo gule je spôsobená použitou transmisívnou BRDF. Situácia

nastala po tom, ako sa lúč, ktorý sa zalomil do vnútra gule, štyrikrát vnútri zrkadlovo odrazil a nakoniec sa vyhodnotila transmisívna BRDF. Tomuto artefaktu sa dá predísť iba znížením použitej hĺbky ray tracingu (zdroj: autor).

4.3.4 Spájanie transmisívnej a difúznej BRDF

To, aby materiál vedel súčasne svetlo lámať a aj difúzne odrážať je fyzikálny nezmysel. Je to preto, lebo obe tieto formy spracovania svetla počítajú s tým, že svetlo prenikne pod povrch, ako to bolo už povedané aj na záver kapitoly 2.1.2, ktorá pojednávala o BRDF ako takej. Táto situácia je v rámci tejto práce veľmi jednoducho ošetrená. Ak má materiál difúzne zafarbenie, teda d z Tabuľka 2 je nenulové, tak materiál automaticky svetlo, ktoré sa neodrazí spekulárne, odrazí difúzne. Ak by zložka d bola nulová, tak by bolo svetlo naopak lomené. V reči BRDF to znamená, že pre každý materiál sa vyhodnocuje buď difúzna BRDF alebo transmisívna BRDF, nikdy nie obe pre ten istý materiál.

4.4 Úloha materiálov pri výpočte zrkadlových odrazov

V kapitole 3.1, v ktorej bolo rozoberané spracovávanie primárnych lúčov počas algoritmu ray tracingu, bolo spomenuté aj vytváranie sekundárnych lúčov, ktoré vznikajú po odraze alebo po lome na rozhraní dvoch materiálov. Každý lúč, ktorý je odrazený, alebo lomený je počas tejto situácie ochudobnený o určitú časť energie, ktorú nesie, pričom toto „ochudobnenie“ má na svedomí materiál, na ktorý tento lúč narazil. V tejto kapitole bude preto popísané, ako vypočítať hodnoty ochudobnenia, nazývané váhy, pre odrazené a lomené lúče na daných materiáloch. Po zvyšok kapitoly uvažujme o vektore v ako o smere lúča, ktorý dopadá na povrch a o vektore l ako o smere odrazeného, respektíve lomeného lúča.

4.4.1 Váha pre zrkadlovo odrazený a lomený lúč

Výpočet váhy pre zrkadlovo odrazený lúč vychádza zo vzorca popísaného v [3]:

$$weight_s(l) = \frac{|lh|F(v,n)G(v,l,h,n)}{|ln||hn|} \tag{4.11}$$

Výpočet váhy pre zrkadlovo odrazený lúč je veľmi podobný a slúži k nemu ďalší vzorec prebratý z [3]:

$$weight_t(l) = \frac{|lh|(1-F(v,n))G(v,l,h,n)}{|ln||hn|} \tag{4.12}$$

Kompletné odvodenie týchto vzorcov sa nachádza v spomínanom článku. Za vysvetlenie stojí úloha vektoru h v tomto vzorci. Ten sa využíva iba v prípade, že v aplikácii je ošetrené vytváranie lesklých odrazov respektíve lomov pomocou distribuovaného ray tracingu, tak, ako to je popísané v kapitole 3.2. Vtedy je totiž na

výpočet tohto odrazu potrebná množina vektorov $l_1, \dots, l_N$. Ich získanie prebieha v dvoch krokoch:

- Najprv získame množinu vektorov $h_1, \dots, h_N$, ktoré predstavujú normály mikroplošiek okolo globálnej normály n. Toto nie je triviálna úloha a bude jej venovaná nasledujúca kapitola 4.4.2.
- V druhom kroku použijeme vzorec (3.1) na výpočet množiny zrkadlovo odrazených lúčov $l_1, \dots, l_N$, respektíve vzorec (3.2) na výpočet množiny lomených lúčov $l_1, \dots, l_N$.

Ak by v aplikácii neprebiehal výpočet lesklých odrazov a lomov na báze distribuovaného ray tracingu, tak by sa namiesto vektorov $h_1, \dots, h_N$ použil len vektor n. To by viedlo k značnému zjednodušeniu vzorcov (4.11) a (4.12).

4.4.2 Vzorkovanie mikroplošiek okolo normály n

Ako bolo spomenuté v prechádzajúcej kapitole, získavanie vzoriek mikroplošiek, respektíve ich normál, je kľúčové pre výpočet lesklých odrazov. Vzorkovanie normál je úzko späté s distribučnou funkciou mikroplošiek popísanou v 4.2.2, nakoľko podľa [3] vzorkovanie normál je vlastne vzorkovaním funkcie $D(h,n)|hn|$. Vzorkovanie, ako je bežné pri vzorkovaní smeru, vracia polárne súradnice navzorkovanej normály h. Vzorkovací algoritmus pre distribučnú funkciu mikroplošiek GGX je prebratý z [3] a má nasledovný tvar:

$$\theta_h = arcta\,n\left(\frac{m\sqrt{\xi_1}}{\sqrt{1-\xi_1}}\right), \phi_h = 2\pi\xi_2 \tag{4.13}$$

V tomto vzorci sú ξ_1, ξ_2 dve náhodne zvolené reálne čísla z intervalu [0,1) a m je drsnosť povrchu predstavená už v Tabuľka 2, ako vlastnosť konkrétneho materiálu. Použitie vzorkovania normály mikroplôšky podľa vzorca (4.13) však vyúsťuje vo výsledných obrazoch v nečakané artefakty. Riešením je nahradiť vzorkovanie normály podľa vzorca (4.13), ktoré odpovedá distribučnej funkcii mikroplošiek GGX, za vzorec na vzorkovanie normály, ktorý by odpovedal Beckmannovej distribučnej funkcii mikroplošiek vyjadrenej vzorcom (4.2). Vzorkovanie pre Beckmannovu distribučnú funkciu je vyjadrené nasledujúcim vzorcom:

$$\theta_h = arctan\left(m^2\sqrt{-log\,(1-\xi_1)}\right), \phi_h = 2\pi\xi_2 \tag{4.14}$$

Použitie Beckmannovho vzorkovania pre normály mikroplošiek (4.14) spolu s distribučnou funkciou mikroplošiek GGX (4.3) výrazne redukuje artefakty spôsobené používaním vzorkovania normály mikroplošiek GGX (4.13), pričom doteraz neboli pozorované žiadne nežiaduce vplyvy tejto zámeny. Porovnanie výsledkov je spolu s popisom znázornené na Obrázok 15.

Obrázok 15: V oboch prípadoch použitá distribučná funkcia mikroplošiek GGX. Hore vzorkovanie normály prispôsobené pre GGX – rovnica (4.13), dole vzorkovanie prispôsobené pre Beckmannovu distribučnú funkciu mikroplošiek – rovnica (4.14) (zdroj: autor).

4.5 Úloha materiálov pri výpočte nepriameho osvetlenia

Pri výpočte nepriameho osvetlenia popísaného v kapitole 2.2.2 aj kaustík v kapitole 2.2.3 je potrebných viacero informácií o materiáloch v scéne, sú to:

- Albedá každého druhu, teda hodnoty ρ_d, ρ_s, ρ_t. Popis, čo to albedo znamená je v kapitole 2.1.3. Tieto sú potrebné pre fázu photon tracingu a budú osobitne rozobrané v nasledujúcich podkapitolách.
- Na samotný odhad nepriameho osvetlenia sa potom používa rovnica (2.10), ktorá používa difúznu BRDF materiálov. Ako ju vypočítať však už bolo ukázané v kapitole 4.3.2, preto táto téma nebude viac rozoberaná.

4.5.1 Výpočet spekulárneho albeda

Výpočet spekulárneho albeda pre materiály sa z veľkej časti riadi postupmi navrhnutými v [5], jediný rozdiel je v použitej váhovej funkcii v kroku 3, nakoľko použitá funkcia (4.11) sa líši od funkcie na počítanie váhy pre zrkadlovo odrazené

lúče, ktorú navrhuje používať Szirmay v [5]. Postup na výpočet spekulárneho albeda prebieha v týchto krokoch:

1. Na začiatku je svetelný lúč l prichádzajúci na povrch s normálou n a parametrom m, ktorý vyjadruje drsnosť povrchu. Situácia sa dá predstaviť ako na Obrázok 7, akurát so zamenenými vektormi v a l. V tomto prípade sa teda vektor l odráža do priestorového uhlu okolo vektoru v.
2. Následne s využitím vzorca (4.14) získame N vektorov $h_1, \dots, h_N$ okolo normály n, z ktorých pomocou vzorca (3.1) vypočítame zrkadlovo odrazené vektory $v_1, \dots, v_N$.
3. Výsledné spekulárne albedo vznikne použitím nasledujúceho vzorca:

$$\rho_s(l,n) = \frac{1}{N}\sum_{i=1}^{N} weight_s(v_i) \tag{4.15}$$

4.5.2 Výpočet difúzneho albeda

Výpočet difúzneho albeda je uvedený v [5] a podľa tohto článku je implementovaný aj pre potreby tejto práce. Vzorec na jeho výpočet je nasledovný:

$$\rho_d(l,n) = d(1 - \rho_s(l,n)) \tag{4.16}$$

4.5.3 Výpočet transmisívneho albeda

Výpočet transmisívneho albeda nie je uvedený v žiadnych referenčných materiáloch. Jeho výpočet by však mohol prebiehať úplne rovnako, ako výpočet spekulárneho albeda popísaného v kapitole 4.5.1, pričom jediný rozdiel by tvorilo lámanie lúčov užitím vzorca (3.2) namiesto ich zrkadlového odrážania.

Inou možnosťou je brať väčší ohľad na zákon zachovania energie a na to, že difúzne albedo a transmisívne albedo by mali byť podobné súdiac podľa kapitoly 4.3.4, kde je popísané, že materiál nemôže súčasne vykazovať difúzne aj transmisívne vlastnosti. V tomto prípade by transmisívne albedo tvorila celá časť svetla, ktorá sa neodrazí zrkadlovo. Ďalšou výhodou takéhoto prístupu je aj to, že nebude dochádzať k žiadnym stratám energie lúču, ktorý dopadne na takýto povrch. Transmisívne albedo sa teda spočíta pre potreby tejto práce jednoducho ako:

$$\rho_t(l,n) = 1 - \rho_s(l,n) \tag{4.17}$$

5. Diskusia

Pri vývoji aplikácie založenej na poznatkoch z predchádzajúcich kapitol došlo k viacerým situáciám, kedy bolo potrebné správne voliť určité pomocné parametre na vykresľovanie výstupných obrazov. To nie je triviálna úloha a nakonfigurovanie aplikácie tak, aby bol výsledný obraz čo najvernejší, ale aby čas spotrebovaný aplikáciou nebol príliš dlhý prebiehalo formou skúšania, simulovania a následného porovnávania výsledkov. Na základe týchto porovnaní bola aplikácia nakoniec vyformovaná do výsledného stavu. Preto budú jednotlivé konfiguračné parametre diskutované osobitne v nasledujúcich kapitolách.

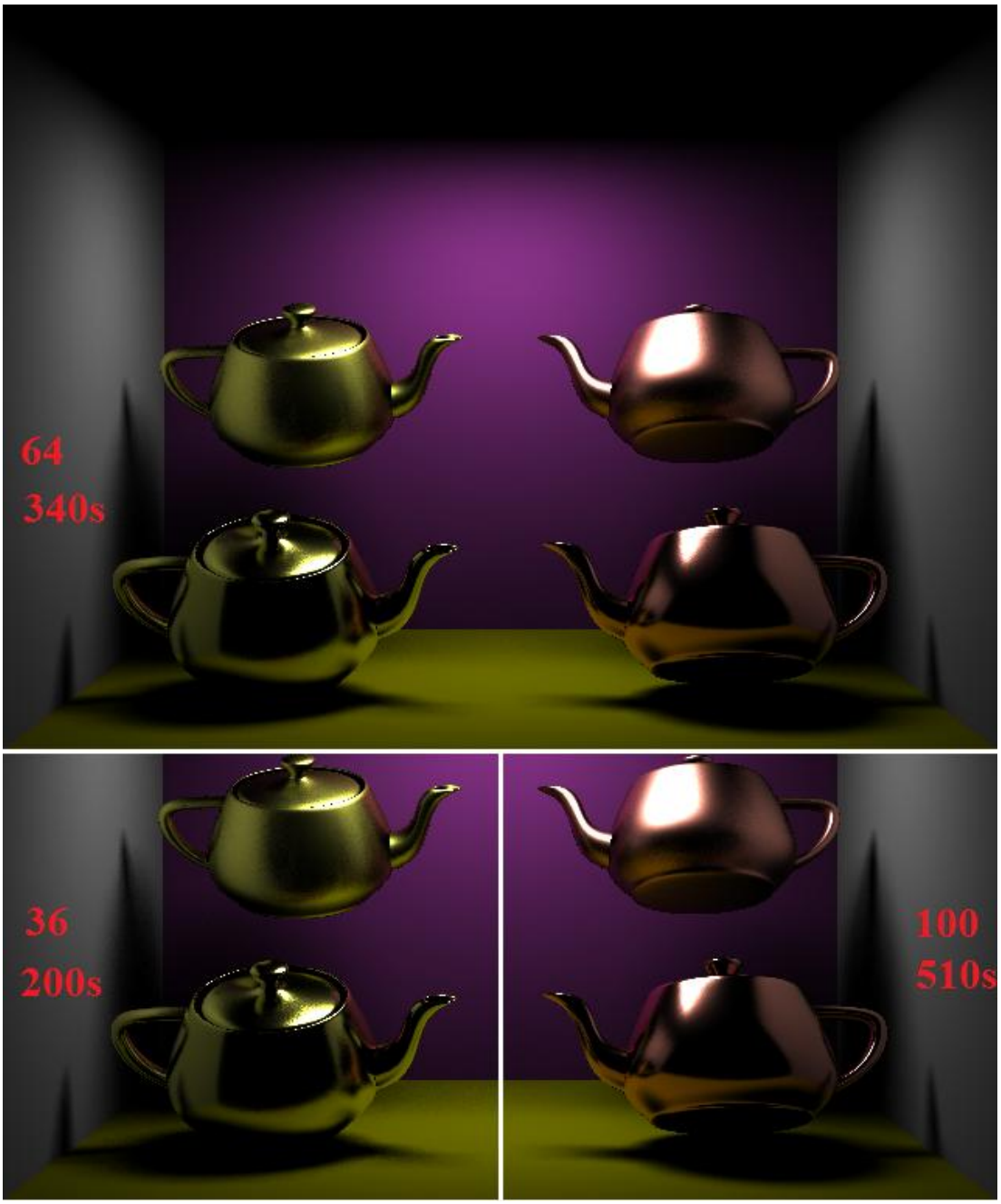
V nasledovnom texte budú často využívané aj časové údaje o dĺžke behu aplikácie pre tú ktorú hodnotu konfiguračného parametra, a preto je na tomto mieste vhodné uviesť konfiguráciu počítača, na ktorom boli testy prevádzané. Jedná sa o počítač s procesorom Intel Core 2 Duo T8300 s frekvenciou každého jadra 2,4 GHz. Aplikácia je zatiaľ schopná bežať len v jednom vlákne, a preto je údaj o počte vlákien irelevantný.

Je vhodné upozorniť, že definície scén vykreslených na obrázkoch v nasledujúcich podkapitolách sa nachádzajú v konfiguračných súboroch na priloženom disku CD, ktorého obsah je popísaný v kapitole 8. Obrázky, z ktorých boli vytvorené demonštratívne koláže a aj samotné koláže pre túto kapitolu sa nachádzajú na disku CD v zložke *images/demo<číslo dema>*. Je to preto, lebo rozdiely v kvalite týchto obrazov sú omnoho lepšie viditeľné v elektronickej, neskreslenej podobe.

5.1 Počet tieňových lúčov – demo1

V kapitole 2.2.1, kde bol popísaný výpočet priameho osvetlenia bol vo vzorci (2.9) spomenutý parameter N, ktorý vyjadroval to, koľko tieňových lúčov sa má použiť pri výpočte priameho osvetlenia v scéne. Obrázok 16 demonštruje výsledok pre rôzne hodnoty tohto parametru, pričom nakoniec bola v aplikácii zvolená hodnota $N = 64$, nakoľko ponúkala evidentné zlepšenie oproti nižším hodnotám a poskytovala rozumný časový kompromis z ponúkaných možností. Vytvorená aplikácia je limitovaná tým, že za hodnotu N mohli byť dosadzované len druhé mocniny celých čísel, teda postupne 1, 2, 4, 9, 16, 32, ... čo prispievalo k rovnomernejšiemu rozmiestňovaniu vzoriek tieňových lúčov.

Na zrýchlenie techniky tieňových lúčov sa dá použiť technika tieňových fotónov, ktorá bude popísaná v kapitole 6.1. Po následnom zrýchlení tejto techniky je možné zvýšiť konštantu N, nakoľko sa ušetrí istý výpočtový čas, a tým pádom táto technika nepriamo prispeje aj k zlepšeniu kvality výsledku.

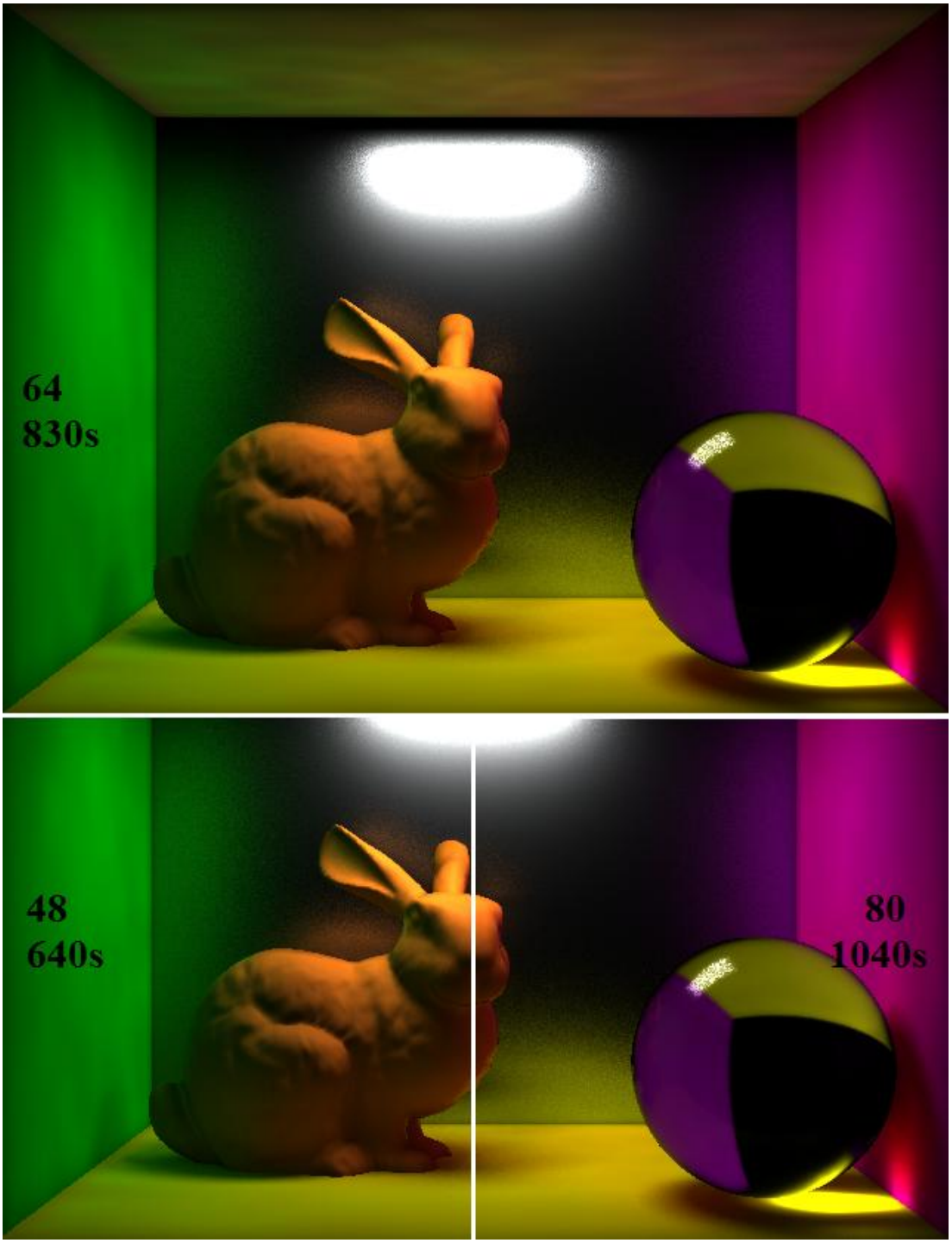


Obrázok 16: Demonštrácia vplyvu počtu tieňových lúčov na kvalitu výstupu a čas potrebný na jeho produkovanie. Počet tieňových lúčov je vyjadrený vrchným číslom a spotrebovaný čas je vyjadrený spodným číslom. Nepriame osvetlenie je z obrázku vynechané úplne, aby vyniklo priame osvetlenie. (zdroj: autor).

5.2 Počet lúčov pre výpočet lesklých odrazov – demo2

V kapitole 3.2 bol popísaný spôsob, akým dosiahnuť vo výsledných obrazoch efekt lesklých odrazov a lomov. Znova bol na tomto mieste použitý parameter N, ktorý predstavoval počet lúčov, ktoré je potrebné na povrchu telesa odraziť, alebo zalomiť na dosiahnutie lesklého efektu. Voľba padla opäť na $N = 64$, i keď ani nie pre oslnivé výsledky, ktoré vzniknú za použitia tejto hodnoty, ale jednoducho preto, že pre vyššie hodnoty sa už výpočet stáva neúnosne pomalý. Najhoršia situácia by

nastala napríklad na sklenenom povrchu, na ktorom by sa 64 lúčov odrazilo a ďalších 64 by sa zalomilo, nakoľko sklo vykazuje aj odrazivé aj lámavé vlastnosti. Demonštrácia vplyvu hodnoty N na kvalitu lesklých odrazov je na obrázku nižšie.



Obrázok 17: Demonštrácia počtu lesklých lúčov na kvalitu výstupu a čas potrebný na jeho produkovanie. Počet lesklých lúčov je vyjadrený vrchným číslom a spotrebovaný čas je vyjadrený spodným číslom (zdroj: autor).

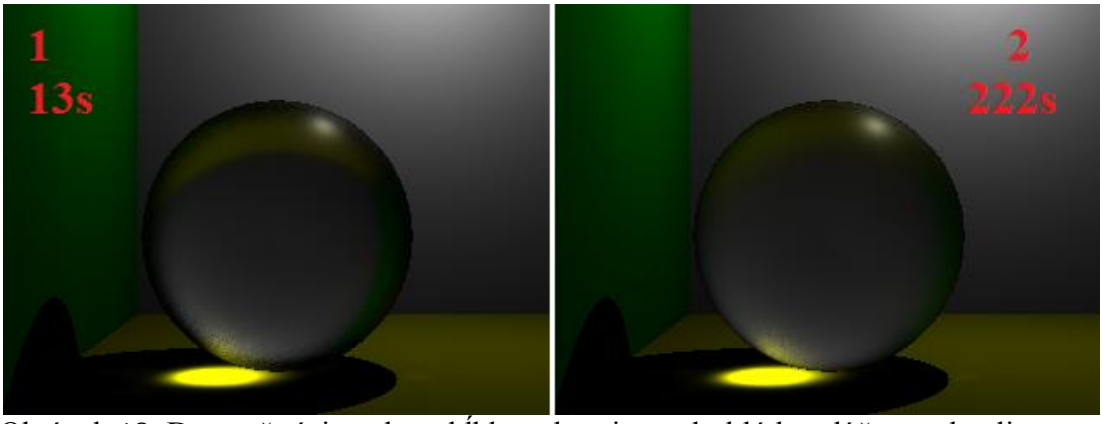
Výpočet lesklého efektu je možné urýchliť a vylepšiť po implementovaní techniky adaptívneho vzorkovania do aplikácie, ktorá bude popísaná v kapitole 6.1.

5.3 Riadenie hĺbky rekurzie ray tracingu – demo3

Riadenie hĺbky rekurzie pre ray tracing patrí k tým najrozšírenejším technikám na zníženie časovej náročnosti aplikácií určených na vykresľovanie užitím metódy ray tracingu. V rámci tejto práce sú zadefinované dve obmedzenia z tejto oblasti:

- Riadenie hĺbky, v ktorej sú maximálne vytvárané lesklé lúče po dopade lúča na povrch telesa. Momentálne je aplikácia nakonfigurovaná tak, aby sa lesklé lúče vytvárali len po dopade primárnych lúčov na povrchy telies.
- Absolútne riadenie rekurzie. Po tom, ako bola prekonaná hranica pre tvorbu lesklých lúčov sa aj naďalej po dopade na teleso vytvára lúč v ideálne odrazenom smere, respektíve ideálne lomený. Aj túto hĺbku je však potrebné riadiť, nakoľko pri nej môže dôjsť k zacykleniu. Pre túto hranicu však neexistuje presne stanovená hodnota, ktorá by bola vhodná pre všetky druhy scén, no kvôli časovej úspore je v rámci tejto práce nastavená len na hodnotu o jedna väčšiu ako hodnota na riadenie hĺbky vytvárania lesklých odrazov.

Demonštrácia vplyvu prvého bodu na kvalitu výsledku je zobrazená na obrázku nižšie.



Obrázok 18: Demonštrácia vplyvu hĺbky rekurzie pre lesklý lom lúčov na kvalitu výstupu a čas potrebný na jeho produkovanie. Hĺbka rekurzie na vytváranie lesklo lomených lúčov je vyjadrená vrchným číslom a spotrebovaný čas je vyjadrený spodným číslom (zdroj: autor).

Ako je evidentné z obrázku, zvýšenie úrovne na vytváranie lesklo lomených lúčov vyústi do radikálneho zvýšenia kvality produkovaného obrazu, no takéto zvýšenie má veľmi dramatický efekt na spotrebovaný čas. Obrázok vyššie obsahuje len primitívnu scénu vykreslenú v nízkom rozlíšení, jediný bodový svetelný zdroj a neobsahuje výpočty nepriameho osvetlenia, iba kaustiky. Aj napriek tomu je výpočtový čas neprimerane dlhý a len podčiarkuje fakt, že zvyšovanie úrovne pre výpočet lesklo odrazených alebo lomených lúčov je bez zásadného urýchlenia celej aplikácie v praxi nepoužiteľné. Naproti tomu, ako je zobrazené napríklad na Obrázok 16, vytváranie lesklo odrazených lúčov len pre primárne lúče je plne dostačujúce pre pekné a verné zobrazenie drsných zrkadlových povrchov.

Demonštrácia druhého obmedzenia je viditeľná napríklad na Obrázok 17, kde zrkadlová stena umiestnená za sklenenou guľou sa javí pri pohľade cez túto guľu ako úplne čierna. Je to preto, lebo lúče, ktoré by boli touto stenou odrazené, by už prekročili stanovenú hĺbku rekurzie.

5.4 Riadenie hĺbky na odhad nepriameho osvetlenia

Tento parameter, respektíve obmedzenie nie je vo fotorealistickej grafike veľmi bežný, no autor tejto práce sa ho rozhodol aj napriek tomu zaviesť, aby tak obmedzil výpočtový čas aplikácie, ktorý by bol spotrebovaný na výpočet nepriameho osvetlenia, ktoré sa spočítavalo aj po dopade sekundárnych lúčov na povrchy telies. Je to preto, lebo v aplikácii ešte nie je implementované efektívne odhadovanie nepriameho osvetlenia, ktoré by malo za následok značnú časovú úsporu. Stručne je podstata tejto vylepšujúcej techniky zhrnutá v 6.1 pod názvom cachovanie nepriameho osvetlenia.

Dopad tohto obmedzenia môže byť veľmi jasne pozorovaný napríklad na Obrázok 17, kedy farba stien, ktoré sú viditeľné cez sklenenú guľu je pomerne tmavšia od ich skutočnej farby, nakoľko obraz v sklenenej guli je ochudobnený o nepriame osvetlenie.

5.5 Riadenie hĺbky výpočtu spekulárnej BRDF – demo4

Vyhodnocovanie spekulárnej BRDF aj pri dopade sekundárnych lúčov na povrchy telies produkuje veľké množstvo šumu vo výsledných obrazoch. Ten by mohol byť v budúcnosti odstránený použitím rovnakej techniky, ktorá bola spomenutá už v kapitole 5.2, teda adaptívneho vzorkovania, ktoré však ešte nie je súčasťou aplikácie.

Existuje však aj iná možnosť ako k tomuto problému pristupovať. Pri výpočte váh pre lomené a odrazené lúče užitím vzorcov (4.11) a (4.12) sa totiž v aplikácii používajú predpočítané hodnoty pre Fresnelov člen F a takisto pre samozatieňovací člen G_1. Tieto predpočítané hodnoty sú v operačnej pamäti udržované pre určité množstvo rôznych uhlov v rozpätí 0°-90°. Čím viac je ale týchto hodnôt uložených v pamäti, tým presnejšie sú počítané váhy pre jednotlivé odrazené a lomené lúče. Vzťah medzi počtom uložených hodnôt a kvalitou výstupu je zobrazený na Obrázok 19.

Bohužiaľ ani druhé navrhnuté riešenie nedokázalo odstrániť problém, dokonca ani pre absurdne veľké počty hodnôt uložených v pamäti, ako je zrejmé zo spomínaného obrázku. Preto autor tejto práce jednoducho odstránil vyhodnocovanie spekulárnej BRDF pre sekundárne lúče, kým nebude súčasťou aplikácie nejaké konečné riešenie tohto problému. Toto rozhodnutie, ako sa ukázalo, nemalo až taký dramatický dopad na úroveň kvality výstupu a pôsobí menej rušivo ako silný šum viditeľný na Obrázok 19. Rovnaká scéna vykreslená užitím tohto zákazu je zobrazená na Obrázok 20, pričom do scény bola náročky pridaná zlatá guľa, ktorá má veľmi silnú spekulárnu BRDF a tak je jej odstránenie pre sekundárne lúče pomerne viditeľné v odraze na hliníkovej konvici.

Obrázok 19: Demonštrácia vplyvu počtu predpočítaných vzorkov pre členy F a G_1 na kvalitu váhy odrazených lúčov. Počet vzorkov je vyjadrený číslom priamo v obrázku, výpočtový čas sa nemenil, no nároky na operačnú pamäť sa so zvyšujúcim počtom zvyšovali (zdroj: autor).

Obrázok 20: Demonštrácia vplyvu odstránenia výpočtu spekulárnej BRDF pri vyhodnocovaní sekundárnych lúčov. Výsledok pred odstránením je na Obrázok 19 (zdroj: autor).

6. Záver

Na začiatku tejto práce stál zápočtový program na predmet Programovanie v C++ bez nejakých väčších ambícií. Postupne však bola aplikácia obohacovaná, vylepšovaná a urýchľovaná, až dospela do súčasnej podoby bakalárskej práce spolu s týmto vypracovaným textom.

Tento text rozdelil problematiku fotorealistického vykresľovania do viacerých logicky oddelených celkov, pričom každému z nich bola osobitne venovaná patričná pozornosť v jednej z predošlých kapitol. Výnimku tvorila len kapitola 1, teda úvod, kde bola vysvetlená potreba a užitie fotorealizmu v počítačovej grafike a takisto bol zhruba načrtnutý smer, ktorým sa táto práca bude uberať, a čo všetko bude obsahovať. Následne v kapitole 2 už bolo rozobraté globálne osvetľovanie, respektíve jeho výpočet na akomkoľvek mieste v 3D priestore, pričom hlavne výpočet globálneho osvetľovania je to, na čo sa v podobných prácach v súčasnosti kladie najväčší dôraz. Inými slovami to znamená navrhnúť postup, ako vypočítať výslednú farbu na povrchu nejakého telesa a brať pri tom do úvahy všetky možné typy osvetlenia, ktoré tento povrch prijíma, vrátane osvetlenia nepriameho. K týmto výpočtom boli uvedené potrebné vzorce a takisto bol načrtnutý postup, ako sa tieto vzorce implementujú v praxi, vrátane teoretického úvodu do techniky fotónových máp, ktorá bola v rámci tejto práce takisto implementovaná za účelom spracovania nepriameho osvetlenia v scénach. Nasledujúca kapitola sa zas venovala problému, ako pixel za pixelom vytvárať výsledné obrazy užitím metódy ray tracingu a teda pretaviť vypočítané globálne osvetlenie do výsledného obrazu. V tej istej kapitole bola poskytnutá aj kriticky potrebná urýchľovacia technika, keďže časová náročnosť vykresľovacích techník založených na algoritme ray tracingu je hlavným kameňom úrazu podobných aplikácií. V kapitole štyri boli rozobraté hlavné špecifiká tejto práce, teda to, ako vypočítať hodnoty potrebné na dosadenie do vzorcov na výpočet globálneho osvetlenia.

Po implementácii všetkých techník, postupov a algoritmov popísaných v kapitolách 2, 3 a 4 vznikla pomerne rozsiahla a variabilná aplikácia schopná produkovať výstupné obrazy scény špecifikovanej užívateľom s komplexne spočítaným globálnym osvetľovaním. Tým pádom bolo možné pristúpiť k produkovaniu výstupných obrazov, hodnoteniu ich kvality, použitiu rôznych konštánt používaných v rámci aplikácie a k ich následnej optimalizácii. Táto časť bola popísaná v kapitole 5 a tým pádom boli naplnené všetky ciele stanovené pred vypracovaním tejto práce. Aj napriek tomu však má výsledná aplikácia implementovaná na základe tohto textu mnoho nedostatkov a zanecháva stále množstvo priestoru pre budúce vylepšenia. Niektoré z nich, plánované v blízkej budúcnosti, sú popísané v nasledujúcej podkapitole.

6.1 Budúce vylepšenia

Tieňové fotóny

Táto technika je spomenutá okrajovo v [1] a slúži na zjednodušenie, respektíve urýchlenie výpočtu priameho osvetlenia, na ktoré sa obyčajne používa technika tieňových lúčov popísaná v kapitole 2.2.1. Základom tejto techniky je to, že dokáže

v scéne identifikovať a rozlíšiť zóny, ktoré sa nachádzajú v tieni, polotieni a ktoré nie sú zatienené vôbec. Výpočet viditeľnosti pomocou tieňových lúčov by tak prebiehal len v oblastiach, ktoré by boli identifikované ako oblasti polotieňa, nakoľko zvyšné oblasti by buď boli plne osvetlené alebo plne v tieni.

Adaptívne vzorkovanie

Adaptívne vzorkovanie pomáha vylepšovať kvalitu a znižovať čas potrebný napríklad na výpočet lesklých odrazov a lomov, ktoré sú popísané v kapitole 3.2. Táto technika funguje na takej báze, že na získanie lesklého obrazu sa nevysiela do priestoru vždy konštantný počet lesklých lúčov. Lúče sa totiž vysielajú do priestoru postupne, pričom sa sleduje nakoľko sa výsledky nimi poskytnuté odlišujú. Ak je odchýlka veľká, tak sa vyšlú ďalšie lesklo odrazené lúče, ak je odchýlka v istej tolerancii, tak sa vysielanie lesklých lúčov ukončí.

Cachovanie nepriameho osvetlenia

Technika opäť navrhnutá v [1], pod názvom irradiance caching, určená na vylepšovanie a urýchľovanie výpočtu nepriameho osvetlenia za použitia techniky fotónových máp. Ak je použitá, tak nie je potrebné nepriame osvetlenie počítať stále nanovo. Spočíta sa iba na niektorých miestach scény, pričom hodnoty nepriameho osvetlenia na týchto miestach sa uložia do pamäte. Následne, ak je potrebné odhadnúť nepriame osvetlenie na nejakom mieste v scéne, tak sa najskôr prehľadá pamäť, či sa z už uložených informácií nedá nepriame osvetlenie aproximovať, namiesto presného výpočtu, ktorý je výpočtovo drahý.

7. Literatúra

[1] Henrik Wann Jensen, *Realistic Image Synthesis Using Photon Mapping*.: AK Peters, 2001.

[2] Matt Pharr and Greg Humphreys, *Physically Based Rendering: From Theory to Implementation*.: Morgan Kaufmann, 2004.

[3] Bruce Walter and others, "Microfacet Models for Refraction through Rough Surfaces," in *Eurographics Conference*, 2007.

[4] Vlastimil Havran. (2000) Heuristic Ray Shooting Algorithms. [Online]. http://dcgi.felk.cvut.cz/home/havran/DISSVH/dissvh.pdf

[5] Laszlo Szirmay-Kalos and Csaba Kelemen, "A Microfacet Based Coupled Specular-Matte BRDF Model with Importance Sampling," in *Eurographics Conference*, 2001.

[6] Robert L. Cook and Kenneth E. Torrance, "A Reflectance Model for Computer Graphics," in *SIGGRAPH '81 Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, 1981.

[7] Istvan Lazanyi and Laszlo Kalos Szirmay, "Fresnel Term Approximations for Metals," in *WSCG*, 2005.

[8] Christophe Schlick, "An Inexpensive BRDF Model for Physically-based Rendering," , 1994.

[9] Petr Beckmann and Andre Spizzichino, *The Scattering of Electromagnetic Waves from Rough Surfaces*.: MacMillan, 1963.

[10] B. G. Smith, "Geometrical shadowing of a random rough surface," in *IEEE Transactions on Antennas and Propagation.*, 1967, pp. 668-671.

8. Príloha A – Obsah priloženého disku CD

Súčasťou tejto práce je disk CD, ktorý obsahuje všetky relevantné súbory súvisiace s touto prácou, ale napríklad aj prácu samotnú. Obsah je rozdelený do viacerých podpriečinkov, a preto bude na tomto mieste rozobratá adresárová štruktúra tohto priloženého disku, spoločne s popisom daných adresárov.

- *demos*: Obsahuje niekoľko vzorových konfiguračných vstupných súborov pre priloženú aplikáciu. To, ako sa narába s týmito vstupnými súbormi je popísané v kapitole 9 a doplňujúce informácie konkrétne k demám sú v kapitole 9.6.
- *doc*: Obsahuje text tejto práce vo formáte pdf.
- *executable*: Obsahuje spustiteľný súbor aplikácie, ktorá implementuje postupy popísané v tejto práci, spolu so všetkými ďalšími súbormi, ktoré táto aplikácia potrebuje k svojmu behu. Užívateľská a programátorská dokumentácia sú k dispozícii v nasledujúcich kapitolách.
- *images*: Táto zložka obsahuje všetky obrázky, ktoré boli použité v rámci tejto práce, nakoľko väčšina z nich je v elektronickej podobe omnoho kvalitnejšia.
- *models*: Táto zložka obsahuje súbory zložitejších objektov, ktoré sa používajú v ukážkových konfiguračných súboroch v zložke *demos*.
- *project*:
    - *photon mapping*: V tomto priečinku sa nachádzajú zdrojové, projektové a iné súbory, z ktorých bola aplikácia v zložke *executable* vytvorená. Zložka predstavuje projekt pre Microsoft Visual Studio 2010, ktoré bolo pri tvorbe aplikácie použité.
    - *include*: Obsah tohto priečinku je potrebné zlúčiť s rovnomenným priečinkom, ktorý používa Microsoft Visual Studio 2010, inak nebude možné projekt zo zložky *photon mapping* preložiť použitím spomenutého Microsoft Visual Studia. Viac o tejto problematike v kapitole 10.2.

9. Príloha B - Užívateľská dokumentácia

Obsah disku CD priloženého k tejto práci tvorí aj spustiteľný súbor aplikácie, ktorá bola vyvinutá na základe postupov popísaných v úvodných kapitolách tejto práce. Slúži teda na vykresľovanie fotorealistických obrázkov, ktoré vzniknú na základe údajov z konfiguračného súboru, ktorý aplikácii poskytuje užívateľ.

V tejto prílohe bude preto popísané, ako sa s touto aplikáciou pracuje. Hneď na úvod je ale vhodné spomenúť, že táto aplikácia zatiaľ beží len pod systémom Windows, v ktorom bola aj vyvíjaná.

9.1 Inštalácia

Aplikácia nevyžaduje žiadnu inštaláciou, pričom tvorí obsah zložky „executable“ na priloženom disku CD. Táto zložka môže byť skopírovaná na ľubovoľné médium a aplikácia bude okamžite schopná behu, pričom sa spúšťa pomocou spustiteľného súboru „pm.exe“, ktorý sa v spomínanej zložke takisto nachádza. Tento však nie je sám o sebe schopný činnosti a k svojmu behu nutne potrebuje všetky súbory zo zložky „executable“. Preto je potrebné dbať, napríklad pri kopírovaní tohto priečinka, aby sa niektorý z týchto súborov pri kopírovaní niekam nestratil.

9.2 Spustenie aplikácie

Ak by bol súbor „pm.exe“ spustený priamo, tak by sa len zobrazilo chybové hlásenie, že aplikácii neboli poskytnuté správne parametre na príkazovom riadku. Aplikácia k svojmu fungovaniu totiž nutne potrebuje konfiguračný XML súbor, pričom cesta k tomuto súboru sa poskytuje vo forme parametra príkazového riadku. Ak bude tento parameter nastavený správne a aplikácia opätovne spustená, tak sa spomínané chybové hlásenie už nezobrazí. To, ako má vyzerať správny konfiguračný súbor je popísané v ďalšej podkapitole.

9.3 Konfiguračný súbor

K pochopeniu toho, ako majú vyzerať správne formátované vstupné súbory pre aplikáciu je potrebná aspoň základná znalosť jazyka XML Schema. Je to preto, lebo pravidlá, podľa ktorých sa tieto vstupné súbory tvoria, sú napísané práve v tomto jazyku. Sú dostupné v súbore „input_file_schema.xsd“, ktorý sa nachádza, tak ako spustiteľný súbor, v priečinku „executable“ na priloženom disku CD.

Samotné pravidlá, respektíve popis XML súboru, sú do patričnej miery okomentované, a preto by bežný užívateľ jazyka XML Schema nemal mať problém po nahliadnutí do spomínaného súboru s pravidlami sám vytvoriť správne a zmysluplne formátovaný vstupný súbor pre aplikáciu.

9.4 Beh aplikácie

Po spustení aplikácie so správnymi parametrami začnú bez ďalšieho užívateľovho zasahovania prebiehať výpočty potrebné k získaniu výsledného obrázku. Prebiehajú v jednotlivých fázach, konkrétne napríklad načítanie vstupného súboru, postavenie

urýchľovacej štruktúry pre vykresľovanie, a tak podobne. Prostredníctvom výstupu do konzoly sa užívateľovi zobrazujú informácie o týchto fázach. Konkrétne kedy tá ktorá fáza začala, skončila a aký dlhý čas bol potrebný na jej vykonanie.

Počas behu programu môže dôjsť k viacerým chybám. Sú to však takmer výlučne chyby spôsobené nesprávne formátovaným vstupným súborom. Text chyby, ku ktorej došlo sa užívateľovi takisto zobrazí do konzoly a aplikácia je následne ukončená.

9.5 Výstup aplikácie

Výstupom aplikácie, ak všetko prebehne bez problémov, je obraz scény, ktorú užívateľ popísal v konfiguračnom súbore. Tento výsledný obraz bude uložený na disk na miesto, ktoré je takisto zadané pomocou konfiguračného súboru. Ohľadom tohto zadania sa ale môže objaviť niekoľko nepríjemností, a preto sa im na tomto mieste bude venovať dodatočná pozornosť.

Dôležitým faktom je, že cesta, ktorú užívateľ zadefinuje v konfiguračnom súboru musí existovať, teda inými slovami, aplikácia nevytvára štruktúru podpriečinkov, ale priamo len výstupný súbor. Ďalším dôležitým faktom je to, že ak súbor s daným menom na danom mieste už existuje, tak bude prepísaný týmto novým výstupným súborom.

Netreba zabúdať ani na formát výstupného obrázku. Ten je špecifikovaný priamo ako prípona výstupného súboru, napríklad „bmp“. Ak je v konfiguračnom súbore zadeklarovaná prípona, ktorú aplikácia nepozná, tak nastane výnimka a aplikácia skončí. Zoznam rozpoznávaných formátov je plne závislý od použitej externej knižnice DevIL[1], preto je na tomto mieste len uvedená prevzatá tabuľka možných formátov a príslušných prípon z dokumentácie danej knižnice, viď Tabuľka 3.

9.6 Vzorové konfiguračné súbory – demá

K aplikácii je na disku CD priložených niekoľko vzorových vstupných konfiguračných súborov, s ktorými si užívateľ môže vyskúšať prácu s aplikáciou. Je však potrebné dať si pozor na niekoľko vecí:

- Momentálne sú demá nakonfigurované tak, že výsledný obraz chcú vytvárať v priečinku, kde sa nachádza spustiteľný súbor. Kým sa ale tento nachádza na disku CD, tak by zápis nebol možný. Preto je potrebné tieto cesty buď prispôsobiť, alebo skopírovať aplikáciu na disk počítača.
- Niektoré z týchto dém, respektíve väčšina, používajú modely, ktoré sa nachádzajú v zložke *models* na priloženom disku CD. V prípade manipulácie so spustiteľným súborom aplikácie je nutné cesty k súborom s týmito modelmi upraviť.

[1] Voľne šíriteľná knižnica na spracovávanie obrázkov, http://openil.sourceforge.net/

Saving	
Type	**Extension(s)**
Windows Bitmap	.bmp
C-style header	.h
DirectDraw Surface	.dds
Jpeg	.jpg, .jpe, .jpeg
Palette	.pal
ZSoft PCX	.pcx
Portable Network Graphics	.png
Pnm	.pbm, .pgm, .ppm
Raw Data	.raw
Silicon Graphics	.sgi, .bw, .rgb, .rgba
Targa	.tga
TIF	.tif, .tiff

Tabuľka 3: Tabuľka prípustných formátov výstupu a príslušných prípon prebratá z dokumentácie knižnice DevIL.

10. Príloha C - Programátorská dokumentácia

Použité algoritmy a postupy už boli popísané v hlavnej časti tejto práce, a preto sa táto časť bude venovať hlavne konkrétnym implementačným riešeniam, ako sú napríklad použité dátové štruktúry, vzťahy medzi triedami a ostatným implementačným špecifikám. Najzákladnejšou informáciou o aplikácii je to, že je celá napísaná v štandardnom jazyku C++, za pomoci niekoľkých externých knižníc, i zdrojového kódu poskytnutého iným autorom.

Súčasťou priloženého disku CD je aj pripravený projekt pre Microsoft Visual Studio 2010, v ktorom bola aplikácia vyvíjaná. V nasledujúcej časti budú preto uvedené informácie potrebné k otvoreniu, kompilácii a spusteniu tohto projektu.

10.1 Spustenie projektu

Samotný projekt sa nachádza na disku CD priamo v zložke „photon mapping“ a tvorí súčasť rovnomenného solution-u, ktoré sa otvára priamo prostredníctvom súboru „photon mapping.sln“. Po otvorení tohto súboru môže existovať stále niekoľko prekážok pred následnou kompiláciou, a to, že táto aplikácia vyžaduje dve externé knižnice, pri ktorých je možné, že ich užívateľ ešte vo svojom Microsoft Visual Studiu nemá. Týmto knižniciam sa venuje nasledujúca podkapitola.

10.2 Potrebné externé knižnice

Aplikácia počas svojho behu vykonáva dve činnosti, pre ktoré už boli vytvorené voľne prístupné multiplatformné knižnice, preto autor uprednostnil ich použitie pred implementáciou vlastných nástrojov. Tieto činnosti sú:

- načítavanie vstupného súboru vo formáte XML, následné overenie správnosti jeho obsahu a interpretácia dát,
- zápis výstupného obrázku do súboru v rôznych formátoch

Na riešenie vyššie spomenutých úloh slúžia konkrétne po poradí knižnice:

- Xercesc[1], jedná sa o veľmi dobre spracovanú knižnicu na tento účel, pričom je dostupná pre rôzne programovacie jazyky. Pre kompiláciu projektu je ale samozrejme potrebná jej verzia pre jazyk C++,
- DevIL, už spomínaná knižnica na manipuláciu s obrázkami v rôznych formátoch. Slúži len pre jazyk C++, no rovnako ako Xercesc, je multiplatformná. V aplikácii sa využívajú konkrétne dve z troch podknižníc tejto knižnice, a to „DevIL“ a „ILu“.

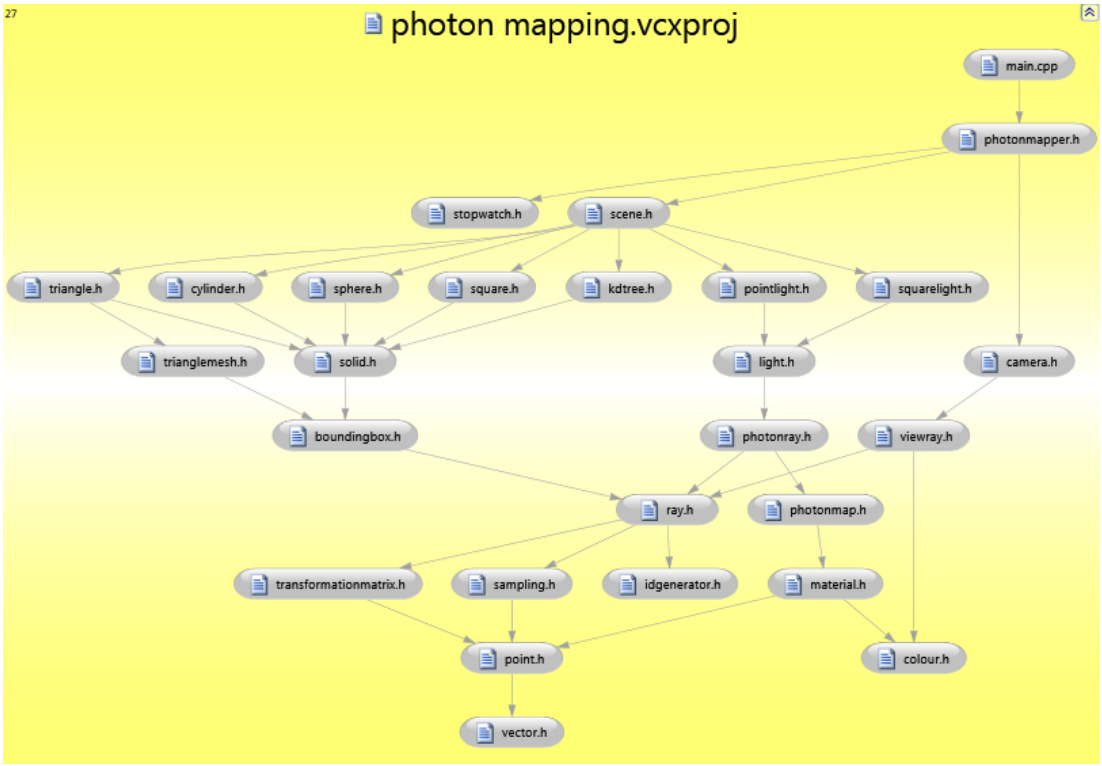
Projekt je napísaný tak, že počíta s tým, že príslušné hlavičkové súbory pre tieto knižnice sa nachádzajú na určitom mieste, a to konkrétne v jednom z podpriečinkov inštalačného priečinku vášho Microsoft Visual Studia. Štandardne je cesta k tomuto podpriečinku nasledovná „Microsoft Visual Studio 10.0\VC\include“. Do neho je potom potrebné umiestniť hlavičkové súbory tak, ako si ich užívateľ

[1] Voľne šíriteľná knižnica na parsovanie súborov vo formáte XML, http://xerces.apache.org/

stiahne zo stránok príslušných knižníc, teda premiestniť sem celé priečinky s názvami „IL“ a „xercesc“. Všetko ostatné je už zariadené.

10.3Projekt

Projekt sa skladá z viacerých súborov obsahujúcich implementáciu jednotlivých tried a jednej výnimky, súboru „input_file_schema.xsd“, ktorý slúži na účely popísané v kapitole 9.3. V tejto kapitole budú popísané jednotlivé triedy, ich zameranie a vzťahy medzi nimi. Často bude využívaný aj Obrázok 21, ktorý zobrazuje diagram závislostí medzi jednotlivými hlavičkovými súbormi.



Obrázok 21: Diagram závislostí medzi jednotlivými hlavičkovými súbormi projektu.

Dajú sa z neho pomerne jednoducho vysledovať aj základné vzťahy medzi triedami, ktoré sú v projekte implementované, nakoľko takmer každá trieda má svoj vlastný hlavičkový súbor (.h) a súbor s kódom (.cpp).

Vstupný bod celej aplikácie sa nachádza v súbore main.cpp, pričom tento súbor neobsahuje takmer nič, tvorí iba vstupnú bránu k zaujímavejším položkám, ktoré budú postupne podľa dôležitosti popísané v nasledujúcich podkapitolách. Pri čítaní nasledovných kapitol je pre lepšie pochopenie odporúčané mať zároveň otvorené aj zdrojové súbory, nakoľko v komentároch, ktoré obsahujú, je veľké množstvo informácií, a preto nie všetko, čo je tam napísané bude znova spomenuté aj v tejto dokumentácii. Ďalej už nasleduje popis jednotlivých tried a niektorých ich kľúčových členov.

10.3.1 Trieda PmPhotonMapper

Táto trieda tvorí jadro celej aplikácie, pričom jej inštancia je vytvorená v tele funkcie main. Nasledovať bude popis jej najvýznamnejších členov:

- `PmPhotonMapper` – jedná sa o konštruktor tejto triedy, ktorý na vstupe dostáva jediný parameter, a to cestu ku konfiguračnému súboru. Ten, ako už bolo povedané, je vo formáte XML, a práve v tomto konštruktore sa využíva vyššie spomínaná knižnica Xercesc. Konštruktor priamo spolupracuje so súborom input_file_schema.xsd, pretože pred samotným načítaním dát a inicializovaním členských premenných prebehne validácia vstupného dokumentu voči tomuto súboru pravidiel jazyka XML Schema. Po úspešnom vykonaní tohto konštruktoru, teda vytvorení inštancií všetkých telies, svetiel, materiálov, či kamery, je všetko pripravené na vykresľovanie výstupného obrázku.
- `distributeLight` - jedná sa o metódu, ktorá na základe popisu scény vyšle zo svetelných zdrojov fotóny a následne zaznamenáva ich prelet scénou, až kým nie sú pohltené. Ak fotóny počas svojho letu dopadnú aj na difúzny povrch, tak je táto situácia zaznamenaná do členskej premennej photonMap. Je privátna, nakoľko ju volá len konštruktor triedy PmPhotonMapper a keďže sa premenná so scénou počas behu programu nemení, tak nie je potrebné ani nikdy viac túto metódu volať, pretože sa tým pádom nemení ani osvetlenie v rámci scény.
- `writeOutput` – táto metóda má na starosť vykreslenie výsledného obrazu na základe dát vytvorených v konštruktore. Obraz produkuje pixel za pixelom a na konci sa pokúsi takto získané dáta zapísať do výstupného súboru. Na tomto mieste je potrebná knižnica DevIL, ktorá sa o tento zápis postará. Potrebná bola jej vyššia verzia, ILu, kvôli lepšej chybovej diagnostike, pretože samotný DevIL počas ukladania obrazu nevie vrátiť text chyby v človeku zrozumiteľnom jazyku a potrebuje na to metódu iluErrorString, ktorá sa o tento preklad postará. O generovanie úvodných lúčov sa postará členská premenná typu PmCamera. Popis tejto triedy viď nižšie.

V zdrojových súboroch, kde je táto trieda implementovaná, sa nachádza aj niekoľko dôležitých konštantných hodnôt, vo forme „define“. Priamo súvisia s metódou writeOutput, pričom sú riadne okomentované. Preto si prípadný záujemca môže nastaviť tieto hodnoty podľa svojich potrieb a prípadne tak zvyšovať alebo znižovať kvalitu výstupného obrazu za cenu predĺženia alebo skrátenia času potrebného na beh aplikácie.

10.3.2 Trieda PmCamera

Jednoducho, priamočiaro implementovaná kamera, ktorú si človek môže predstaviť ako fotoaparát v bežnom svete. Užívateľ ju pomocou konfiguračného súboru nastavuje, umiestňuje, ... Na „pohyb“ po scéne využíva transformačnú maticu.

Jej hlavnou úlohou v programe je vytvárať inštancie triedy PmViewRay, inými slovami, kamera vytvorí lúč s počiatkom vo svojom ohnisku a koncom vo zvolenom bode na svojej priemetni. O túto úlohu sa stará metóda getRay, ktorá na vstupe dostáva polohu spomínaného bodu na priemetni kamery.

10.3.3 Trieda PmScene

Trieda obsahujúca všetky informácie o scéne, ktorú má aplikácia vykresliť. Ako sa spomína aj v zdrojovom texte, obsahuje popis telies a svetiel v scéne, no takisto aj popis materiálov v scéne použitých. Tieto spomenuté údaje sa v tejto triede ukladajú vo forme inštancií tried PmSolid, PmLight a PmMaterial. Jej najdôležitejšími členskými funkciami sú:

- `getVisibility` – tá zisťuje, či sa medzi dvoma bodmi v priestore nachádza nejaká prekážka, alebo či sú priamo „viditeľné“. Táto metóda sa používa v aplikácii výlučne len pri odhade priameho osvetlenia istého miesta v scéne pomocou tieňových lúčov.
- `getClosestIntersection` – úplne najkľúčovejšia metóda z pohľadu metódy sledovania lúča. Pre konkrétny lúč s počiatkom a smerom zistí, či zasahuje nejaký objekt v scéne, a ak áno, tak dopočíta aj všetky ostatné vlastnosti potrebné na ďalšie spracovanie tohto prieniku.
- `getLighting` – metóda, ktorá využíva už vyššie spomenutú getVisibility. S jej pomocou vypočítava osvetlenie určitého bodu v scéne, a to ako difúznu zložku, tak aj spekulárnu, či transmisívnu.
- `getEmittedPhotonRays` – posledná zaujímavá metóda. Pre účely metódy mapovania fotónov, ktorá sa v aplikácii používa na odhadovanie nepriameho osvetlenia. Jej účelom je vygenerovať určitý počet fotónov zo všetkých svetelných zdrojov v scéne, pričom počet fotónovej pre to ktoré svetlo závisí od jeho celkového výkonu.

Dôležitou členskou premennou tejto triedy je premenná typu PmKDTree. Používa ju ako metóda getVisibility, tak i metóda getClosestIntersection na urýchľovanie.

10.3.4 Triedy PmLight a PmSolid

Jednotné rozhranie pre typy svetelných zdrojov, ktoré sa môžu v scéne nachádzať, je popísané formou abstraktnej triedy PmLight. Čiže každý typ svetelného zdroja je jej priamym potomkom, pre príklad viď triedy PmSquareLight, alebo PmPointLight. Jedným z nedostatkov tohto rozhrania, ako i celej aplikácie, je zatiaľ to, že svetelné zdroje sa vo výslednom obrázku nezobrazujú. Je to preto, že rozhranie PmLight neprikazuje implementovať metódu na zisťovanie prieniku, a tak sa takáto metóda ani nikde na príslušných miestach v aplikácii nevyvoláva.

Na tomto mieste by sa mohol nachádzať analogický komentár, ako pre triedu PmLight, avšak s tým rozdielom, že by reč nebola o rozhraní svetelných zdrojov, ale o rozhraní telies, ktoré môžu byť v scéne prítomné, a to tak, že implementujú abstraktnú triedu PmSolid. Tu už samozrejme nemôže byť reč o chýbajúcej metóde počítajúcej prienik lúča a telesa, nakoľko táto metóda telies scény predstavuje gro celého vykresľovania ako takého, nielen v tejto konkrétnej aplikácii. Pre príklady už implementovaných tried telies sú k nahliadnutiu triedy ako PmSquare, PmSphere a iné.

Pre potenciálnych záujemcov o rozšírenie aplikácie o ďalšie druhy svetelných zdrojov, alebo telies sú práve tieto dve triedy správnym miestom, kde začať. Vďaka komentárom v zdrojovom texte by následná tvorba vlastných, nových typov svetelných zdrojov a telies nemala spôsobovať akékoľvek problémy. Druhou stranou

mince je však to, že implementovať novú triedu nestačí. Aby sa vôbec v aplikácii dala použiť, je potrebné do XML schémy konfiguračného súboru pridať pravidlá povoľujúce obsah nového typu, a takisto konštruktor triedy PmPhotonMapper si bude musieť vedieť s takýmto záznamom z konfiguračného súboru nejako poradiť. Užívateľ však môže čerpať inšpiráciu z už implementovaných svetelných zdrojov a telies, ktoré sú v aplikácii plne podporované.

10.3.5 Trieda PmKDTree

Táto trieda už bola spomenutá v kapitole o 10.3.3, kde bol takisto krátko spomenutý jej hlavný účel. Tým je urýchľovanie hľadania prienikov lúčov a scény, no i bez toho je PmKDTree samo o sebe zaujímavou triedou. Ako je už z názvu zrejmé, implementuje známu, a v počítačovej grafike široko využívanú, dátovú štruktúru KD strom, ktorá už bola popísaná aj v hlavnej časti tohto dokumentu v kapitole 3.3.1.

Jej implementácia však obsahuje zopár programátorských špecifík, ktoré na tomto mieste popíšem. KD strom, medzi iným, je popísaný aj pomocou maximálnej hĺbky, ktorú môže dosiahnuť a maximálnym počtom položiek, ktoré môžu obsahovať uzly, ktoré sú listami, inými slovami, ktoré nie sú ďalej rozdelené. Tieto dve konštanty už boli popísané takisto skôr, no stojí zato spomenúť, že potenciálny záujemca o rozšírenie aplikácie sa s ich hodnotami môže jednoducho skúsiť pohrať, pretože sú jednoducho dostupné, nakoľko sú zadefinované hneď na začiatku hlavičkového súboru obsahujúceho deklaráciu triedy PmKDTree.

10.3.6 Trieda PmPhotonMapAdvanced

Dôležitosť tejto triedy ako takej nie je veľmi vysoká, no jej význam spočíva v tom, že každá jej inštancia v sebe ukrýva viacero členských premenných typu PhotonMap. Ako už bolo povedané skôr, aplikácia využíva viacero druhov fotónových máp na skladovanie fotónov rôzneho druhu, pričom práve tieto rôzne fotónové mapy spadajú pod jeden všeobecnejší objekt typu PmPhotonMapAdvanced.

Dôvodom je zjednodušenie prístupu k dátam, ktoré tieto mapy poskytujú, pretože na výpočet osvetlenia z nich všetkých naraz stačí zavolať jedinú metódu triedy PmPhotonMapAdvanced. Ďalším príkladom využitia takéhoto všeobecnejšieho objektu je napríklad to, že v rámci svojich členských premenných obsahuje údaje o tom koľko a v akom okruhu okolo špecifikovaného bodu máme hľadať fotóny pri odhadovaní osvetlenia.

10.3.7 Trieda PhotonMap

Ako už bolo povedané, hlavná funkcionalita fotónových máp, ako boli popísané v hlavnej časti tohto dokumentu sa nachádza v tejto triede a nie v triede PmPhotonMapAdvanced.

V prvom rade je však potrebné upozorniť užívateľa, že zdrvujúca časť tejto triedy, respektíve jej implementácie nepochádza od autora tejto práce. Fotónová mapa tak, ako je implementovaná je prebratá od Henrika Wann Jensena [1].

Na tomto mieste preto uvediem iba dôležité metódy, ktoré boli do tejto triedy pridané autorom, spolu s ich krátkym popisom, nakoľko popis zvyšných metód je k nájdeniu v spomenutej literatúre.

- `getNearestPhotons` – jedná sa o metódu, ktorá má za úlohu zozbierať z fotónovej mapy určitý maximálny počet fotónovej okolo miesta v priestore špecifikovanom užívateľom. Z veľkej časti je táto metóda kópiou metódy locate_photons, ktorá nie je napísaná autorom. Je však upravená tak, aby skutočne našla všetky fotóny v okolí špecifikovaného bodu, pretože v pôvodnej metóde bolo niekoľko fotónov vynechaných. Bližšie informácie sú dostupné v komentári v zdrojovom texte.
- `getLighting` – Jensen v pôvodnej triede Photon_map implementoval len jednoduchú metódu, ktorá spočítala sumu z energie, ktorú niesli jednotlivé fotóny okolo špecifikovaného miesta. Táto metóda však dokáže vypočítať priamo osvetlenie daného bodu v priestore, na základe vlastností materiálu, na ktorom sa spomínaný bod nachádza.

10.3.8 Trieda PmMaterial

V tejto triede sa nachádza najväčšie množstvo obsahu špecifického pre túto aplikáciu. Ide totiž o to, že distribuovaný ray tracing, alebo photon mapping ako taký, je implementovaný v množstve aplikácií. Dôležitejšie a špecifickejšie pre jednotlivé aplikácie je to, ako sa lúče v ray tracingu a fotóny vo photon mappingu odrážajú od telies. Telesá sú samozrejme zložené z materiálov, a tak je logickým vyústením, že táto funkcionalita sa nachádza v tejto triede.

Trieda znova obsahuje niekoľko kľúčových metód, pričom je celá dobre okomentovaná, a preto sa im budem venovať iba veľmi skrátene. Dôležité metódy by sa dali rozdeliť do dvoch skupín:

- Metódy, ktoré vypočítajú a vrátia údaj o tom, aká časť svetla sa odrazí v nejakom bode v priestore, ktorý nie je špecifikovaný svojou polohou, ale uhlami medzi vektormi do svetelného zdroja, do oka pozorovateľa, normálou, a tak ďalej. Týmito metódami sú getDiffuseBRDF, getSpecularBRDF a getTransmissiveBRDF, ktoré sú naimplementované na základe poznatkov zhrnutých v kapitolách 4.3.1, 4.3.2 a 4.3.3.
- Ďalšia skupina metód je zameraná na výpočet koeficientov, ktorými budú prenásobené lúče, ktoré sú od týchto materiálov odrazené alebo nimi lomené. Týmito sú getWeightForReflectedRay a getWeightForRefractedRay, ktoré sú implementované na základe popisu z kapitoly 4.4.1. Do tejto skupiny by sa dali zaradiť aj metódy getTransmittance a getTransmittanceOfThinSurface, ktoré vypočítajú a vrátia údaj o tom, aká časť z lúča bude pohltená po prelete týmto materiálom o určitej hrúbke.