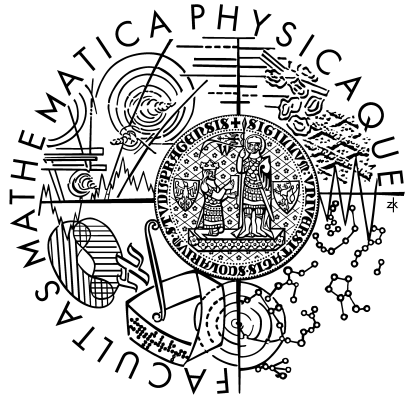
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Zdeněk Kavalír

Spektrální optický simulátor

Kabinet software a výuky informatiky

Vedoucí bakalářské práce: RNDr. Josef Pelikán

Studijní program: Informatika

Studijní obor: programování

2010

Děkuji vedoucímu své práce, RNDr. Josefu Pelikánovi, za jeho čas, cenné připomínky a vedení po celou dobu projektu. Své ženě děkuji za její pochopení a trpělivost, bez které by tato práce nemohla být dokončena.

Prohlašuji, že jsem svou bakalářskou práci napsal(a) samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 24. 5. 2010 Zdeněk Kavalír

Obsah

Název práce: Spektrální optický simulátor
Autor: Zdeněk Kavalír
Katedra: Kabinet software a výuky informatiky
Vedoucí bakalářské práce: RNDr. Josef Pelikán
E-mail vedoucího: Josef.Pelikan@mff.cuni.cz

Abstrakt: Cílem této práce je navrhnout a implementovat systém pro simulaci optických jevů se zaměřením na spektrální zdroje světla a disperzní jevy v optických prvcích vhodný pro didaktické i experimentální použití.

Vstupem pro simulaci je definiční soubor scény ve formátu XML popisující zdroje světla, optické prvky a stínítka, výstupem obraz zachycující jednak světlo dopadající na stínítko, jednak celkový pohled na scénu obsahující optické prvky i světlo procházející soustavou. Zdroj světla může být bodový s uživatelsky definovaným spektrálním složením nebo rastrový daný vstupním bitmapovým obrázkem. Jako optické prvky v definičním souboru scény je možné použít základní geometrická tělesa nebo z nich metodou CSG vytvářet složitější. Definiční soubor scény umožňuje použití aritmetických výrazu a proměnných.

Simulace je založena na principu distribuce světla do scény (*light-tracing*) metodou Monte-Carlo. Chování světla na rozhraní dvou prostředí o různém indexu lomu je řešeno analyticky, nepoužívá se zjednodušení vycházející z chování paraxiálních paprsků.

Klíčová slova: optický simulátor, spektrální rendering, disperze světla, sledování světla, Monte-Carlo

Title: Spectral optical simulator
Author: Zdeněk Kavalír
Department: Department of Software and Computer Science Education
Supervisor: RNDr. Josef Pelikán
Supervisor's e-mail address: Josef.Pelikan@mff.cuni.cz

Abstract: The aim of this thesis is to design and implement an optical simulator focused on spectral light sources and dispersion effects, suitable for educational and experimental purposes.

The input of the simulation is an XML scene definition file describing light sources, optical elements and optical sheets. The result of the simulation is one or more pictures containing incident light footprints collected on optical sheets as well as side sheets observing scene from a distance along with track of light passing through the scene. A light source can be defined as a point light with user-defined spectral distribution or a bitmap light using input bitmap picture as a glowing texture. The shape of any optical element can be defined as a basic solid, or constructed using CSG techniques. The scene definition file specification allows using of arithmetic expressions and user-defined variables.

The simulation is based on a distribution of light from light sources toward the scene using a Monte-Carlo algorithm. Behaviour of light incident onto a transparent boundary is computed precisely according to the geometry of an element; a paraxial ray approximation is not used.

Keywords: optical simulator, spectral rendering, dispersion theory, light tracing, Monte-Carlo

Kapitola 1

Úvod

Spektrální jevy provázejí člověka doslova na každém kroku – v dnešní době je člověk obklopen řadou výrobků ze skla či plastu, které ve slunečních paprscích vytvářejí světelné obrazce plné spektrálních barev. Sama příroda umí vykouzlit úchvatné scenérie, od modré oblohy, červánků při západu Slunce nebo duhou při dešti až po halové jevy vznikající lomy a odrazy na krystalcích ledu v atmosféře.

Další velkou kapitolou jsou optické přístroje, denně se setkáváme s dioptrickými brýlemi, lupami nebo objektivy fotoaparátů, pro astronomy jsou optické soustavy dokonce nezbytnými prostředky výzkumu. Při zobrazování optickými přístroji se spektrální jevy zpravidla nevyskytují, to je ale dáno důmyslnou konstrukcí, která tyto jevy eliminuje (např. achromatické objektivy) – při návrhu optické soustavy musí být spektrální chování řádně ošetřeno.

Neprofesionální zájemci o studium spektrálních jevů jsou zpravidla odkázáni na odborné články nebo hotové přístroje. Proto by bylo zajímavé vytvořit simulátor spektrálních jevů, který by mohl sloužit jako virtuální optická laboratoř – uživatel by mohl definovat světelné zdroje a optické vlastnosti materiálů, sestavit optickou soustavu a sledovat chování světla v interakci s objekty scény.

Spektrální optický simulátor by tak mohl být přínosný pro amatérské astronomy, kteří sestavují dalekohled vlastní konstrukce, pro přípravu doprovodných materiálů k výuce optiky, případně pro konstruktéry optických soustav.

1.1 Cíle práce

Cílem této práce je navrhnout a implementovat systém, který umožní simulaci průchodu světla optickou soustavou s ohledem na spektrální složení světla. Vstupem je definice optické scény popisující zdroje světla a geometrii optických prvků, výstupem pak obraz vykreslený na některém ze stínítek umístěných ve scéně, příp. pohled na celou scénu z pozice vnějšího pozorovatele (v dalším textu označován jako *snímek*

kamery).

V definici optické scény má být možné definovat

- spektrální složení bodového zdroje světla
- rastrový obrázek jako zdroj světla („zářící textura“)
- závislost indexu lomu optického prvku na vlnové délce
- optické prvky technikou CSG modelování
- pozorovací průmětny (stínítka), uložení obrazce zachyceného světla do souboru

Vstupem aplikace by měl být definiční soubor ve formátu XML, vytvořený kód by měl být přenositelný a měl by podporovat paralelní výpočty ve více vláknech.

Vytvořená aplikace může být užitečná při návrhu konstrukce libovolné optické soustavy, ve výuce optiky, ale i při individuálním zkoumání optických jevů a jiných účinků světla.

1.2 Struktura textu

1. Úvod – tato kapitola
2. Teorie – shrnuje teoerické poznatky využité při návrhu a implementaci simulátoru
3. Návrh aplikace – obecný návrh aplikace, definice světelného modelu, vymezení možností modelu scény
4. Implementace – objektový model, použité algoritmy, optimalizace
5. Uživatelská příručka – spuštění programu, popis definičního souboru, interpretace výstupu programu
6. Příručka programátora – postup kompilace, základní orientace v kódu
7. Výsledky – testovací úlohy pro ověření správné funkce simulace, efektivita paralelizace

Kapitola 2

Teorie

Cílem simulace je výpočet obrazce, který vzniká dopadem světla emitovaného světelným zdrojem a procházejícího danou optickou soustavou. Pro realistický výpočet je nutné vycházet z fyzikální podstaty světla a chování světla v interakci s optickými prvky, ze kterých optická soustava sestává. V této kapitole uvedu fyzikální východiska a jevy, které jsem v simulaci uvažoval, ustanovím také matematický model, který zřejmě vymezuje možnosti simulace.

2.1 Optika

Snahy o fyzikální popis chování světla vyústily ve dvě teorie, které se navzájem doplňují, vlnové a korpuskulární. V jistých situacích lze chování světla vysvětlit vlnovou teorií, v jiných naopak vysvětlení podá pouze teorie korpuskulární. Mluvíme tak o vlnově-korpuskulárním dualismu.

Vlnová teorie Světlo je elektromagnetické vlnění (popsané Maxwellovými rovnicemi), které se šíří od zdroje světla; zavádí se pojem vlnoplochy spojující body se shodným fázovým posuvem.

Korpuskulární teorie Energie není stejně a spojitě rozložena, vyskytuje se v jistých centrech — kvantech (tato kvanta nazval A. Einstein fotony), energie kvanta je dána frekvencí přidruženou k fotonu.

2.1.1 Barva světla, spektrální složení

Lidské oko je schopno vnímat světlo o vlnové délce v rozmezí přibližně $380--780nm$, různé vlnové délky jsou subjektivně vnímány jako odlišné *barvy*. V běžných podmínkách je světelné záření složeno z elektromagnetického vlnění o různých vlnových délkách – *spektrální charakteristika* světla pak popisuje závislost intenzity záření obsaženého ve světle na jeho vlnové délce. Optickými přístroji je možné zobrazit

spektrální složení světla, jeho *spektrum* – využívají přitom principu ohybu světla, příp. lomu na hranolu, při kterém dochází k odklonu paprsků v úhlu závislém na vlnové délce; vzniká tak barevný pruh, ve kterém barvy přecházejí jedna do druhé. Barvy, vyskytující se ve spektru se označují jako *spektrální*. Spektrum může být

- *čárové*, je tvořeno navzájem oddělenými spektrálními čarami, vysílají jej např. rozžhavené plyny a páry chemických prvků;
- *spojité*, vytvářejí jej např. rozžhavené látky ve skupenství pevném nebo tekutém, výjimečně plyny při velmi vysokém tlaku (klasická žárovka, sluneční záření); spojité spektrum je ve všech případech stejné, nezávisí na chemickém složení zdroje.

Barva světla je dána jeho spektrálním složením, monochromatické světlo (o jediné vlnové délce) má jednu ze spektrálních barev, všechny ostatní barvy vnímané lidským okem vznikají působením světla se spektrem složeným.

Fraunhoferovy čáry Německý fyzik J. Fraunhofer zjistil, že sluneční spektrum – na první pohled spojité – obsahuje řadu nespojitostí, „černých“ čar. Jak bylo později objeveno, záření o vlnových délkách odpovídajících chybějícím spektrálním čarám je pohlcováno jednak v chladnější okrajové části Slunce (chronosféře), jednak v atmosféře Země. Nejsilnější z těchto čar označil římskými písmeny A-H; toto označení stanoví určité místo ve spektru mnohem přesněji než neurčité označení barevného odstínu.

`Čára`	A	B	C	D	E	F	G	H
$\lambda[nm]$	$760,82$	$686,72$	$656,28$	$589,3$	$527,0$	$486,14$	$430,78$	$396,85$

Závislost indexu lomu na vlnové délce Index lomu optické látky se liší pro různé vlnové délky. Je-li nutné znát index lomu pro libovolnou vlnovou délku, je možné použít např. *Cornu-ův vzorec*,

$$n = n_0 + \frac{a}{\lambda - \lambda_0}, \tag{2.1}$$

v němž a, n_0, λ_0 značí konstanty. K určení těchto konstant stačí, známe-li indexy lomu n_1, n_2, n_3 pro světla vlnových délek λ_1, λ_2, λ_3. Po eliminaci konstant (viz [literatura]) dostáváme

$$n = n_3 + \frac{n_1 - n_3}{1 + N},$$

kde

$$N = \frac{(\lambda_1 - \lambda)(\lambda_3 - \lambda_2)(n_1 - n_2)}{(\lambda - \lambda_3)(\lambda_2 - \lambda_1)(n_2 - n_3)}$$

2.1.2 Geometrická optika

Geometrická optika pracuje s prostými geometrickými pojmy, jako jsou bod, přímka nebo rovina. Světelný zdroj je tvořen svítícími body, ze kterých se světlo šíří paprsky; odtud také jiné označení, *paprsková optika*. Pojem paprsku lze zdůvodnit v obou výše uvedených teoriích – v korpuskulárním modelu odpovídají paprsky drahám jednotlivých částic emitovaných z bodového zdroje, vlnová teorie pak hovoří o paprsku jako o normále k vlnoplochám.

V této práci se omezím jen na případ přímočarého šíření světla, odraz a lom na rozhraní dvou prostředí. Vycházím přitom z následujících předpokladů:

- ve stejnorodém a izotropním prostředí se světlo šíří přímočaře ve tvaru světelných paprsků,
- ve světelném toku jsou jednotlivé paprsky na sobě nezávislé,
- na rozhraní dvou stejnorodých a izotropních prostředí platí pro světelné paprsky zákon lomu a odrazu.

Jakékoliv další vlivy nejsou při zpracování uvažovány, jevy přesahující tyto předpoklady (např. difrakce, interference) nelze v simulaci sledovat.

2.2 Statistika a pravděpodobnost

Při zkoumání jevů, které nastávají bez známé příčiny (např. výsledek hodu hrací kostkou), je často nutné se uchýlit ke kvantitativnímu přístupu a uvažovat četnosti jevů, které v dané situaci mohou nastat. Tento postup vede ke stanovení *pravděpodobnosti* výskytu *náhodných jevů*, coby výsledků *náhodných pokusů*.

2.2.1 Náhodná veličina

Náhodná veličina je zobrazení, které každému elementárnímu (nedělitelnému) náhodnému jevu $\omega_i \in \Omega$ přiřazuje jediné reálné číslo $x \in \mathbb{R}$,

$$X : \omega_i \mapsto X(\omega_i)$$

Diskrétní náhodná veličina Diskrétní n.v. je popis náhodného pokusu, který končí konečně nebo spočetně výsledky $\{x_i\}$ s pravděpodobnostmi $\{p_i\}$ a platí přitom

$$p_i \geq 0, \qquad \sum_i p_i = 1$$

.

Spojitá náhodná veličina Je-li množina výsledků náhodných pokusů nespočetná (např. měření vzdálenosti nebo výšky postavy), hovoříme o spojité náhodné veličině. Rozdělení pravděpodobnosti je dáno *distribuční funkcí*, resp. *hustotou pravděpodobnosti*.

2.2.2 Distribuční funkce

Distribuční funkci k náhodné veličině X definujeme jako

$$F(x) = P(X \leq x), \qquad x \in \mathbb{R},$$

kde $P(X \leq x)$ je pravděpodobnost, že hodnota náhodné veličiny X je menší než x. Za předpokladu $P(\Omega) = 1$ a $P(\emptyset) = 0$ platí, že

- $F(x)$ je neklesající
- $0 \leq F(x) \leq 1$
- $lim_{x \to -\infty} F(x) = 0$, $lim_{x \to \infty} F(x) = 1$

Kvantilová funkce Distribuční funkce plně popisuje rozdělení pravděpodobnosti náhodné veličiny. V praxi je však mnohdy nutné řešit úlohu nalezení hodnoty x tak, aby hodnota $P(X \leq x)$ byla rovna dané hodnotě $\alpha \in (0, 1)$. Tato úloha zřejmě vede k hledání inverzní funkce k distribuční funkci $F(x)$; pro tuto funkci byl zaveden pojem *kvantilová funkce*.

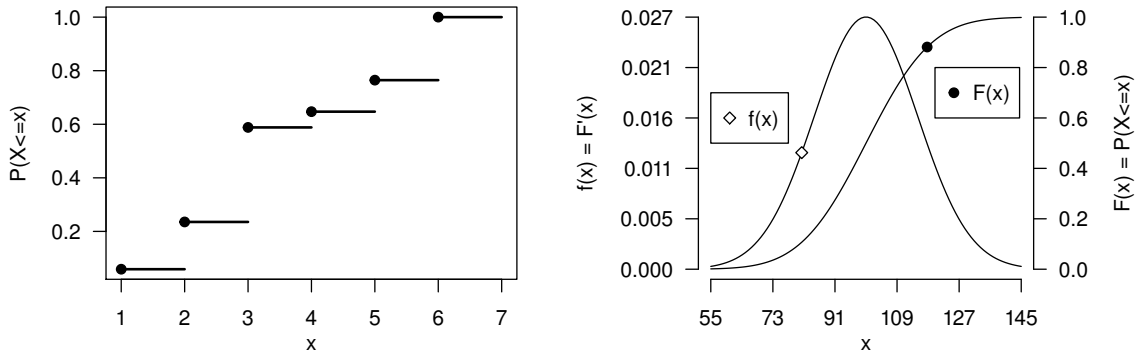
2.2.3 Hustota pravděpodobnosti

Mějme distribuční funkci $F(X)$ spojité náhodné veličiny. *Hustotou pravděpodobnosti* $f(x)$ rozumíme funkci, pro níž platí

$$F'(x) = f(x)$$

Jako důsledek dále platí

- $F(x)$ je nezáporná pro $\forall x \in \mathbb{R}$
- $F(x) = \int_{-\infty}^{x} f(t)dt$
- $\int_{\mathbb{R}} f(t)dt = 1$
- $P(a < X \leq b) = F(b) - F(a) = \int_{a}^{b} f(t)dt$



Obrázek 2.1: a) distribuční funkce diskrétní náhodné veličiny, b) distribuční funkce a hustota pravděpodobnosti *normálního rozdělení* $N(100, 15)$

2.2.4 Metoda Monte-Carlo

Metoda Monte-Carlo je numerická simulační metoda založená na opakovaném provádění náhodných pokusů s cílem statisticky popsat chování zkoumaného systému. Jedná se o metodu značně neefektivní, pro dosažení dostatečné přesnosti je nutné provést velké množství testů – chyba klesá s asymptotou $k/\sqrt{N}$, kde N je počet provedených pokusů. Metoda je tedy vhodná pouze v případech, kdy alternativní metody dávají horší výsledky nebo zcela selhávají.

Výpočet určitého integrálu Velice názornou aplikací metody je úloha výpočtu plochy pod danou reálnou funkcí $f(x)$ – náhodný pokus spočívá ve zvolení bodu (x, y) v rovině s náhodně zvolenými souřadnicemi (s rovnoměrným rozdělením pravděpodobnosti). Porovnáním hodnoty y a $f(x)$ lze rozhodnout, zda daný bod leží uvnitř nebo vně zkoumané plochy. Pravděpodobnost, že bod bude uvnitř plochy (úspěšný pokus), odpovídá obsahu této plochy. Statistickými nástroji lze dokázat, že při dostatečně vysokém počtu pokusů je podíl úspěšných pokusů libovolně blízko této pravděpodobnosti, a tedy i obsahu zkoumané plochy. Touto metodou je možné – s jistou přesností – numericky vyčíslit hodnotu určitého integrálu; uplatnění nachází především v případech, kdy je snadné vyšetřit vzájemnou polohu bodu a dané funkce a naopak analytické řešení je nesnadné – jedná se např. o integrace ve více dimenzích nebo integraci s komplikovanými mezemi.

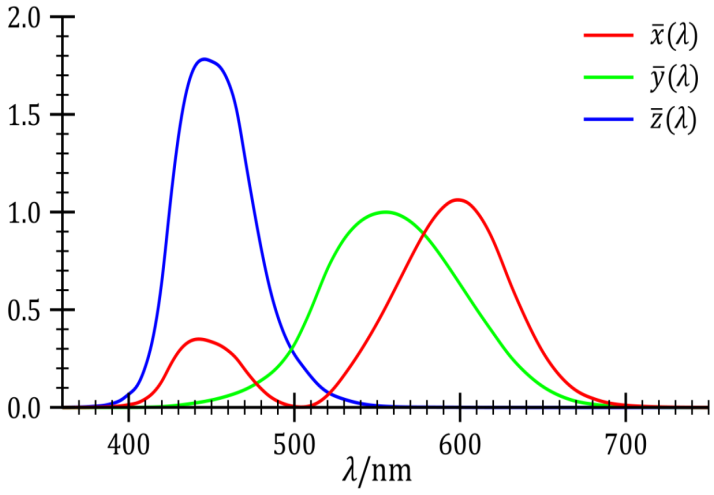
2.3 Počítačová grafika

2.3.1 Barevné prostory

Na sítnici lidského oka se nacházejí dva receptory citlivé na světlo – tyčinky vnímají jasovou složku, čípky zajišťují barevný vjem. Rozlišení jednotlivých barev je možné

díky různé citlivosti čípků na konkrétní vlnové délky; rozlišují se tři typy čípků s nejvyšší citlivostí na barvy odpovídající přibližně červené, zelené a modré barvě.

Barevný prostor XYZ Na základě experimentálního výzkumu J. Guilda a W. D. Wrighta vytvořila komise *Commission Internationale de lEclairage* (dále jen CIE) definici barevného prostoru XYZ. V této normě se předpokládá, že oko vnímá třemi orgány, jejichž citlivost k barvám je vyjádřena křivkami v obr. 2.2. Souřadnice $\overline{x}$, $\overline{y}$, $\overline{z}$ pro určitou vlnovou délku odpovídají barevným podnětům spektrální barvy zkoumané vlnové délky (viz [2]).



Obrázek 2.2: Křivky citlivost oka k barvám (převzato z [5])

S touto definicí (označme křivky jako $\overline{x}$, $\overline{y}$, $\overline{z}$) je možné pro dané světelné záření o známém spektrálním složení $I(\lambda)$získat hodnoty X, Y, Z jako

$$X = \int_0^\infty I(\lambda)\,\overline{x}(\lambda)\,d\lambda, \quad Y = \int_0^\infty I(\lambda)\,\overline{y}(\lambda)\,d\lambda, \quad Z = \int_0^\infty I(\lambda)\,\overline{z}(\lambda)\,d\lambda \tag{2.2}$$

2.3.2 Stochastické metody zobrazování

Při stochastickém zobrazování trojrozměrné scény se vyšetřují cesty paprsků mezi zdroji světla a pozorovatelem (viz [1]). Vychází se přitom ze dvou základních konceptů – sledování scény z pohledu pozorovatele (metoda *sledování paprsku*) a distribuce světla směrem od zdroje do scény (*sledování světla*).

Metoda sledování paprsku Z každého pixelu pozorovací roviny je vyslán paprsek směrem do scény. Je-li nalezen průsečík, v závislosti na vlastnostech povrchu je rozhodnuto o míře příspěvku světla ze směru zrcadlového odrazu, následně se postup rekurzivně opakuje, nebo je rekurze ukončena a vyhodnocen příspěvek zdroje

světla ve směru zkoumaného paprsku. Hodnota výchozího pixelu odpovídá celkovému příspěvku vyslaného paprsku. Metoda výborně zobrazuje zrcadlové odrazy.

Metoda sledování světla Ze zdroje světla je vyslán paprsek a podobně jako v metodě sledování paprsku prochází scénou. V každém průsečíku paprsku se scénou je zkoumán příspěvek do pozorovací roviny – je vyslán *paprsek příspěvku* (contribution ray) směrem k pozorovateli a protíná-li pozorovací rovinu, je hodnota odpovídajícího pixelu nastavena na hodnotu příspěvku. V porovnání s metodou sledování paprsku metoda mnohem lépe zobrazuje *kaustiky* (obrazce vytvořené průchodem světla průhlednými tělesy).

Kapitola 3

Návrh aplikace

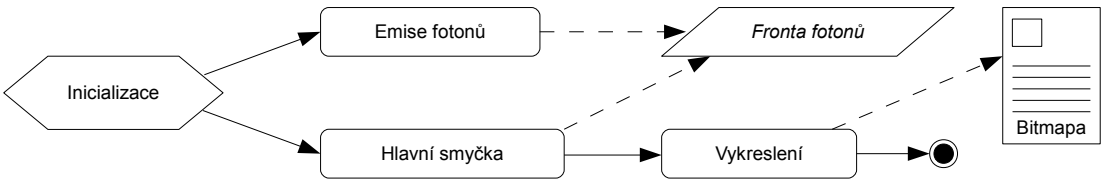
3.1 Základní koncepce

Při simulaci je sledována trasa světla vycházejícího ze zdrojů, výpočet vychází výhradně z geometrického popisu prvků, nejsou zavedena žádná zjednodušení ani aproximace. Je-li kladen důraz na studium barevných účinků světla, bude výpočet – při vhodné definici scény[1] – brát v úvahu spektrální složení světla a odděleně zpracovávat jednotlivé paprsky podle jejich vlnové délky.

Kromě optických prvků může scéna obsahovat také „virtuální“ objekty, které se vykreslují do výstupních obrazců, nejsou však součástí výpočtu – jedná se o měřítka, úhloměry a doplňkové grafické objekty, které mohou být zajímavé pro následnou analýzu výstupního obrazce.

3.1.1 Postup výpočtu

Simulace je založena na distribuci světla do scény, výpočet je tedy řízen zdroji světla, které generují fotony a předávají je ke zpracování scéně. Výpočet se skládá z následujících kroků (viz obr. 3.1):



Obrázek 3.1: Postup výpočtu

Inicializace Je načten definiční soubor scény a sestavena scéna.

[1] je nutné definovat spektrální složení světelných zdrojů a pro každé těleso specifikovat závislost indexu lomu na vlnové délce

Emise fotonů Světelné zdroje emitují fotony a ukládají je do fronty fotonů.

Hlavní smyčka Hlavní smyčka programu zajišťuje zpracování fotonů – vyjímá fotony z fronty a vyhodnocuje interakci se scénou. Zpracování jednoho fotonu může vyvolat vytvoření nových fotonů (např. rozklad světla na rozhraní), které jsou uloženy opět do fronty. Hlavní smyčka končí v okamžiku, kdy jsou všechny fotony zpracovány.

Vykreslení Vyhodnotí se příspěvky fotonů každému stínítku, obraz je převeden do RGB prostoru a uložen do souboru.

3.2 Světelný model

3.2.1 Transport světla

Světlo je emitováno ze zdroje, prochází optickou soustavou a dopadá na stínítko, případně bez užitku opouští scénu. Světelný obrazec vykreslený na stínítku vzniká působením světelného toku Φ, který dopadá na stínítko. Osvětlení E v daném bodě plochy je rovno podílu světelného toku $\mathrm{d}S$ této plochy a velikosti tohoto plošného elementu, číselně je tedy osvětlení rovno světelnému toku, který dopadá na plošnou jednotku osvětlovaného tělesa (viz [2]).

Celkový světelný tok dopadající v bodě plochy je výsledkem integrace světelného toku přes všechny směry poloprostoru nad tímto bodem. Výpočet tohoto integrálu probíhá numericky, metodou Monte-Carlo. Světelný tok každého světelného zdroje je rovnoměrně rozdělen do velkého množství diskrétních částic, které jsou následně emitovány do scény.

Každá tato částice prochází optickou soustavou samostatně, při dopadu na některé stínítko svou energií přispívá k celkové přijaté energii v daném bodě stínítka, po zpracování všech emitovaných částic tyto dopadající částice vytvářejí na každém stínítku výsledný obrazec. Nezávislost průchodu jednotlivých částic scénou je důležitá pro možnost paralelizace výpočtu.

Tyto částice budeme v dalším textu označovat jako *fotony*.

3.2.2 Model fotonu

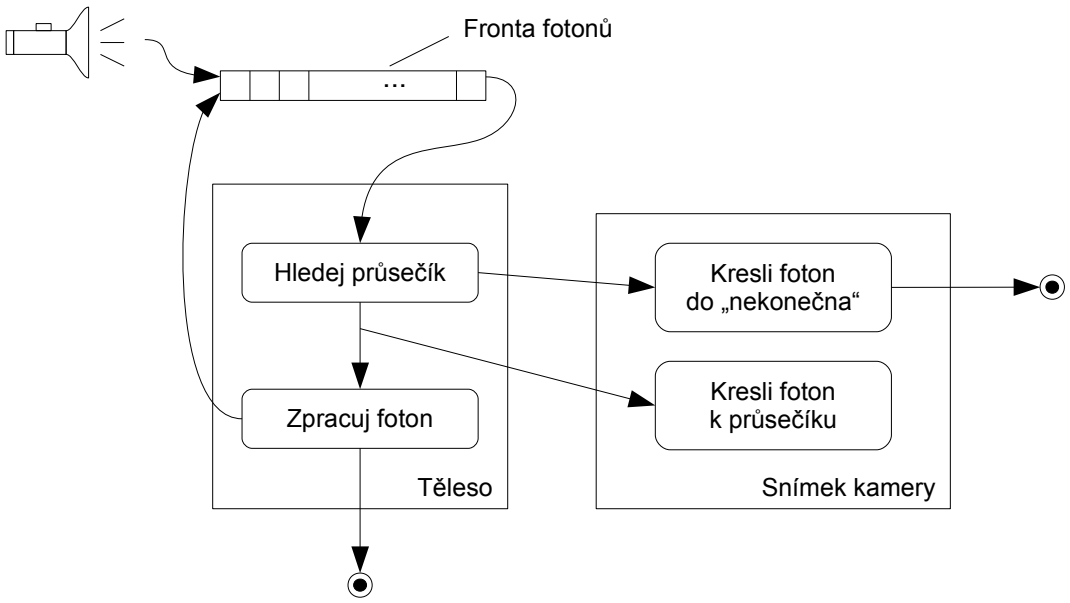
Foton je elementární vzorek světla emitovaný ze zdroje světla, geometricky reprezentuje paprsek světla (tj. polopřímka s počátkem a směrovým vektorem). Foton má v okamžiku emise následující vlastnosti:

- počátek – náhodně zvolený bod na povrchu světelného zdroje
- směrový vektor – náhodně zvolený z předepsaného rozsahu

- energie – zlomek celkové energie zdroje
- barva – v závislosti na definici zdroje daná spektrem, nebo RGB barvou

3.2.3 Životní cyklus fotonu

Fotony vznikají ve světelném zdroji, každý nový foton je umístěn do *fronty fotonů* – je-li fronta naplněna, je emise pozastavena (výpočetní vlákno zdroje je uspáno). *Hlavní smyčka fotonu* (viz obr. 3.2) vyzvedne foton z fronty a hledá nejbližší průsečík se scénou:



Obrázek 3.2: Hlavní smyčka programu

Průsečík neexistuje Není-li žádný průsečík nalezen, je foton zrušen. Je-li definován snímek kamery, ještě před zrušením je na snímek vykreslena stopa fotonu.

Průsečík existuje V opačném případě je foton předán tělesu, na jehož povrchu byl průsečík nalezen, algoritmus zpracování fotonu může zpět do *fronty fotonů* vložit buď tentýž foton bez změny (transparentní povrch, např. stínítko), nebo libovolný počet nových fotonů (lom na rozhraní), příp. žádný foton nevložit a původní foton pouze zrušit (optická past, clona). Je-li definován snímek kamery, je po nalezení průsečíku foton předán k vykreslení na stínítku.

Je-li nalezen průsečík se stínítkem, je na stínítko zaznamenán příspěvek fotonu a foton je vložen s novým počátkem (na povrchu stínítka) zpět do fronty fotonů.

Foton může být ukončen také z důvodu překročení maximálního povoleného počtu interakcí (odrazů nebo lomů na rozhraní) – tento limit je prevencí proti zacyklení

v případě, že se foton dostane do situace, kdy je opakovaně odrážen nebo lomen a nemůže opustit scénu nebo být jinak zrušen.

3.2.4 Barva fotonu

Fotony emitované z nespektrálního zdroje mají RGB barvu (danou v definici scény), tato barva je po celou cestu od zdroje přes optické prvky až do stínítka neměnná.

Spektrální zdroje světla by měly emitovat fotony všech vlnových délek rozdělených shodně s definovanou spektrální charakteristikou. Z důvodu zvýšení efektivity výpočtu jsou ze spektrálních zdrojů světla emitovány fotony *spektrální* a rozhodnutí o vlnové délce fotonu se odkládá až na okamžik, ve kterém je pro výpočet nezbytná – při lomu světla na prvním disperzním[2] optickém rozhraní. Vyšší efektivita výpočtu spočívá ve skutečnosti, že všechny fotony o stejné geometrii procházejí soustavou až do prvního takového rozhraní shodně, bez ohledu na svou vlnovou délku. Místo n fotonů o různých vlnových délkách tak ze zdroje vychází jediný foton o daném spektrálním složení a k distribuci monochromatických fotonů (rozklad světla) dochází později.

Foton tedy může být

- nespektrální, barva je dána RGB hodnotou
- monochromatický, barva je dána (jedinou) vlnovou délkou
- spektrální, barva je dána spektrálním složením

3.2.5 Spektrální rozdělení fotonů

Při lomu na disperzním rozhraní jsou nespektrální a monochromatické fotony zpracovány standardním, výše uvedeným způsobem – je upraven počátek a směrový vektor paprsku, barevné vlastnosti se nemění.

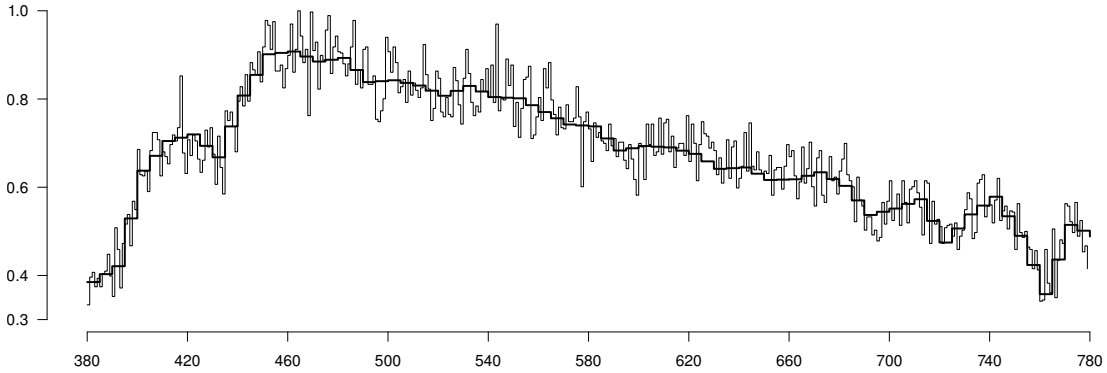
V případě spektrálního fotonu se po dopadu spektrálního fotonu vytváří dané[3] množství nových, monochromatických fotonů s různými vlnovými délkami. Cílem je, aby rozložení vlnových délek při rozkladu světla odpovídalo spektrálnímu složení zdroje světla, tj. histogram vlnových délek fotonů generovaných v daném bodě byl shodný s jeho spektrální charakteristikou. Jelikož není předem možné zjistit počet fotonů, které budou v daném bodě takto vytvořeny, není možné podle spektrální charakteristiky předem určit počet fotonů pro každou vlnovou délku. Kompromisní řešení uchovávat statistiku již vytvořených fotonů by kladlo neúnosné nároky na paměťový prostor a nutně snižovalo nezávislost výpočtu jednotlivých fotonů, a tedy i volnost paralelizace; jedná se tedy o nevhodné řešení.

[2] optický prvek s materiálem `dispersion="yes"`

[3] volitelné v definičním souboru

Jak z předchozí diskuse vyplývá, přesné rozložení vlnových délek nových fotonů je obtížně dosažitelné. Ukazuje se však, že dobrého výsledku lze dosáhnout aproximací – generováním náhodných hodnot vlnových délek s rozložením pravděpodobnosti odvozeným ze spektrálního složení. A protože již pro integraci příspěvků světla na elementárních ploškách stínítek byla zvolena metoda Monte-Carlo, a tedy nutnost emitovat velké množství fotonů (řádově nejméně 10^7), bude zajištěn dostatečný počet fotonů i pro tuto aproximaci.

Algoritmus, jímž se generují hodnoty vlnových délek pro nové hodnoty, je velice přímočarý – každý spektrální zdroj světla ve scéně má definovanou spektrální charakteristiku, a to buď některou z vestavěných (standarní zářič D65, A), nebo uživatelsky definovanou (viz 5.2.3). Tato charakteristika se považuje za hustotu pravděpodobnosti rozložení vlnových délek v daném světle, z ní je sestavena distribuční a kvantilová funkce (inverzní k distribuční). Při vytváření nového fotonu se z generátoru náhodných čísel s rovnoměrným rozdělením pravděpodobnosti generuje číslo, které se chápe jako kvantil spektrálního rozdělení, hledaná nová hodnota vlnové délky je pak získána z kvantilové funkce.. Na obrázku 3.3 je znázorněno srovnání histogramu takto získaného spektra s výchozí spektrální charakteristikou.



Obrázek 3.3: Srovnání distribuční funkce spektrální rozdělení zářiče D65 a histogramu vlnových délek generovaných metodou Monte-Carlo s 10 000 vzorky

3.2.6 Spektrální rozdělení RGB fotonů

Při zpracování fotonu emitovaného z RGB zdroje[4] se pro červenou, modrou a zelenou barvu standardně generují fotony s vlnovou délkou po řadě 700, 546,1 a 460 nm[5]. Pro foton je vytvořeno spektrální rozdělení odpovídající intenzitám r, g, b – při rozkladu světla mají všechny fotony stejnou energii, liší se však počet fotonů – pro složku s vyšší intenzitou bude emitováno více fotonů příslušné vlnové délky.

[4]bitmapový zdroj nebo bodový zdroj s barvou danou RGB hodnotou

[5]hodnoty vlnové délky pro základní barvy je možné změnit v definičním souboru

3.3 Model scény

Všechny objekty scény jsou umístěny v třídimenzionálním euklidovském prostoru, jediná omezení hodnot vektorů a vzdáleností vyplývají pouze z datového typu reálného čísla použitého v konkrétní implementaci. Pozice a tvar každého tělesa je jednoznačně popsán v definičním souboru a v průběhu výpočtu se nemění.

3.3.1 Základní objekty

Objekty scény se rozdělují především podle

- své funkce – zda emitují fotony či nikoli
- viditelnosti pro fotony – zda interagují s emitovanými fotony
- viditelnosti pro kameru – zda se fotony zobrazují na snímku kamery (viz dále v textu)
- optických vlastností – jakým způsobem bude dopdadající foton zpracován
- citlivosti – zda je možné graficky zobrazit fotony dopadnuté na povrch („fotocitlivý“ materiál)

Podle tohoto rozdělení je možné – s ohledem na cíle této práce – vytvořit základní objekty, ze kterých bude sestavena každá scéna. V tabulce 3.4 a následujícím přehledu jsou vymezeny jejich vlastnosti a popsáno chování, které se od nich očekává.

	emise	viditelnost		fotony	citlivost
		fotony	kamera		
zdroj světla	ANO	–	–	–	–
stínítko	–	ANO	–	propouští	ANO
černé těleso	–	ANO	ANO	pohlcuje	–
průhledné těleso	–	ANO	ANO	láme / odráží	–
zrcadlo	–	ANO	ANO	odráží	–
virtuální těleso	–	–	ANO	–	–

Obrázek 3.4: Rozdělení objektů scény podle vlastností a využití

Zdroj světla Světelný zdroj je nehmotný objekt, jenž ze svého povrchu emituje fotony. Může se jednat o zdroj *bodový*, kdy rozměr zdroje je zanedbatelný vzhledem k vzdálenostem ve scéně a složení emitovaného světlo je popsáno svými vlastnostmi, nebo *bitmapový*, kdy fotony jsou emitovány z povrchu obdélníku, na jehož povrch je mapována daná bitmapa a složení světla odpovídá hodnotě bitmapy v bodě, z něhož foton vychází.

Stínítko Těleso, které registruje dopadající fotony a po dokončení výpočtu ukládá vytvořený obrazec jako bitmapu. Fotony stínítkem procházejí bez změny, jeden foton tedy může přispět do obrazce několika stínítkům zároveň.

Černé těleso Optická past, těleso, které každý foton dopadnutý na povrch ukončí bez náhrady; pohlcuje tedy veškeré světlo. Touto svou vlastností je vhodné pro modelování clony omezující světelný svazek, příp. v kombinaci se stínítkem vytváří neprůhledné stínítko, které tak plně odpovídá chování stínítka v simulaci na reálné optické lavici.

Průhledné těleso Těleso, jehož materiál je čirý o daném indexu lomu. Světlo jím volně prochází, v závislosti na úhlu dopadu a indexu lomu je podle Snellova *zákona lomu* lomeno, nebo odraženo.

Zrcadlo Zrcadlový povrch tělesa způsobí, že každý foton je po dopadu na povrch odražen dle zákona odrazu zpět do scény. Dojde-li v nějaké scéně k situaci, ve které je foton opakovaně odrážen mezi dvěma nebo více objekty, je výpočet chráněn před zacyklením nastavením maximálního počtu interakcí fotonu s objekty (je možné nastavit v definici scény) – po dosažení tohoto limitu je foton ukončen.

Virtuální těleso Pro větší názornost obrazců vykreslených na stínítku je možné do scény vložit objekty, které jsou vykresleny ve výstupní bitmapě aniž by jakkoli ovlivnily výpočet – jedná se především o měřítka, úhloměry a konstrukční prvky, které přiblíží výstupní obrazec reálné optické lavici. Definicí virtuálního objektu je bitmapa mapovaná na obdélníkové těleso.

3.3.2 Zobrazení na stínítku

Stínítko je těleso scény ve tvaru obdélníka, jehož povrch je namapován na výstupní bitmapu znázorňující obrazec vykreslený dopadajícím světlem.

Zachycení paprsku Pro operaci hledání průsečíku fotonu se scénou je stínítko běžné těleso, které může zpracovat dopadající foton. Stínítko je průhledné, dopadající foton je bez jakékoli změny vrácen do fronty fotonů a tedy se šíří dále scénou, jako by stínítka nebylo. Má-li být stínítko neprůhledné, stačí do definice scény přidat neprůhledné[6] těleso o stejné geometrii, a to na pozici za definicí stínítka[7]

[6] materiál typu `trap`

[7] překrývají-li se dva objekty ve scéně, jsou zpracovány v pořadí, v jakém jsou uvedeny v definici scény, zde je tedy nejprve foton zpracován stínítkem, pak teprve pohlcen na neprůhledném objektu

3.3.3 Snímek kamery

Snímek kamery se snaží zachytit scénu tak, jak by mohla být spatřena vnějším pozorovatelem v průběhu simulace. Oblast viditelná na snímku je dána pozicí a orientací obdélníku kamery v definičním souboru scény. Snímek zobrazuje pravoúhlou projekci objektů scény a fotonů procházejících danou oblastí.

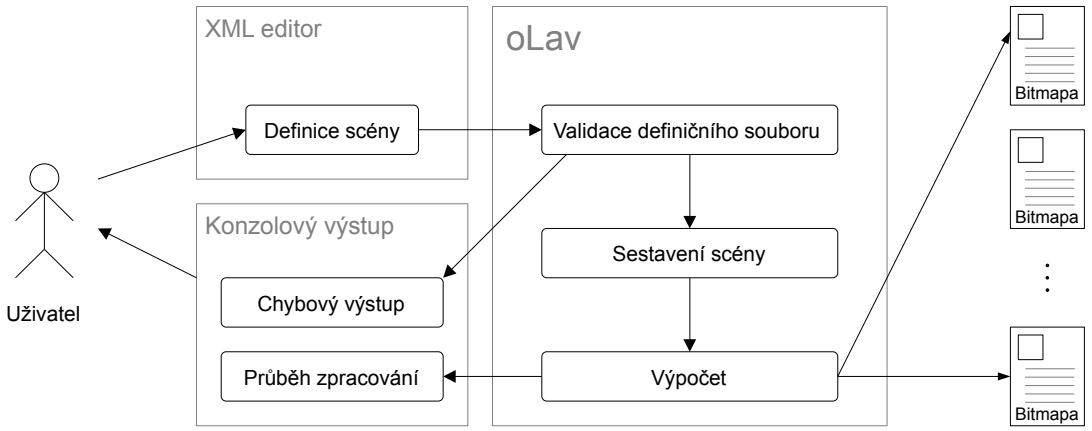
Objekty scény Metodou sledování paprsku (ray-tracing) je z každého pixelu obdélníku snímku do scény vyslán paprsek a je-li nalezen průsečík se scénou, je příslušný pixel snímku obarven barvou tělesa upravenou kosinem úhlu paprsku a normály povrchu v bodě průsečíku.

Stopy fotonů Stopy fotonů procházejících scénou se v žádné podobě neukládají do datových struktur, nemohou být tedy zobrazeny metodou sledování paprsku. Jsou proto na stínítko vykreslovány v průběhu celého výpočtu vždy v okamžiku zpracování fotonu.

3.4 Návrh aplikace

Simulace optické lavice je úloha především výpočetní – ač návrh scény i grafický výstup úzce souvisí s vizuální představou, simulace je založena na aplikaci teoretických poznatků o šíření a povaze světla a nalezení celkového příspěvku světelných zdrojů na elementární plošky stínítek zkoumaných v simulaci.

Aplikace neobsahuje editor definičního souboru scény, interakce uživatele s aplikací se tedy omezuje na pouhé spuštění výpočtu. Proto je zcela dostatečné ovládat práci aplikace z textové konzole, významně se tím také zvyšuje přenositelnost mezi různými platformami.



Obrázek 3.5: Schematické znázornění práce s aplikací

Jak je znázorněno na obr. 3.5, práce s aplikací sestává z vytvoření definičního souboru v externím editoru a spuštění výpočtu. Výstupem aplikace jsou pak soubory s rastrovým grafickým výstupem, nebo chybové hlášení v případě nalezení chyby v definici scény. Externím editorem scény se rozumí autonomní aplikace, jejímž výstupem je dobře formovaný XML soubor se strukturou popsanou v uživatelské příručce, viz kapitola 5.

3.4.1 Paralelizace výpočtu

Pro zvýšení efektivity výpočetních aplikací je vhodné využít možnosti paralelního zpracování, jež současné procesory běžně nabízejí. Simulace optické lavice pracuje s pojmy geometrické optiky, považuje přitom všechny fotony emitované ze zdrojů světla za navzájem nezávislé (viz oddíl 2.1.2); tuto úlohu lze tedy zřejmě snadno paralelizovat.

V návrhu paralelizace vyjděme ze schematu 3.5. Iniciální sestavení scény a finální uložení výstupních souborů není z důvodu množství souborových operací vhodné paralelizovat.

Významný přínos však znamená paralelizace pro hlavní výpočetní smyčku. Podúlohy, které *nemění stav scény* (stav mohou pouze číst) je možné zpracovávat souběžně bez dalších opatření. Jedná se především o následující operace:

- Nalezení průsečíku foton se scénou – objekty scény nejsou hledáním průsečíku nijak ovlivněny, výsledek operace má dopad pouze na foton samotný.
- Řešení odrazu nebo lomu na rozhraní – rozhraní pouze příslušném algoritmu dodává údaje o materiálu tělesa, lomem ani odrazem není nijak dotčeno.

Jiná situace nastává u podúloh, které *stav scény mění*. Zde je nutné vhodným způsobem řídit přístup k objektům scény. Jedná se především o následující operace:

- Emise fotonů ze světelného zdroje – i v případě náhodně generovaných fotonů je sdílen přístup k počítadlu již emitovaných fotonů.
- Vložení a vyzvednutí fotonu z fronty – nedostatečně ošetřený přístup k frontě může ovlivnit správnou funkci a tím znehodnotit celý výpočet.
- Vyhodnocení příspěvku fotonu dopadnutého na povrch stínítka – na stínítku se neukládají jednotlivé fotony, nýbrž přičítá se (barevný) příspěvek fotonu, což zpravidla nebývá triviální, atomická operace.

Způsob řízení paralelního výpočtu bývá závislý na cílové platformě i operačním systému. Pro zajištění přenositelnosti kódu je to nutné mít na paměti a při implementaci vhodným způsobem zohlednit.

Kapitola 4

Implementace

Tato kapitola popisuje vývoj aplikace od volby implementačního prostředí přes objektový model až k podrobnému popisu vybraných komponent a použitých algoritmů a metod.

Aplikace je ryze výpočetní, celou implementací se tedy táhne snaha o efektivitu kódu a přesnost matematických výpočtů. Dalším významným cílem je přenositelnost kódu, jež umožní spouštět úlohu na různých strojích počínaje běžnými osobními počítači až po výkonné výpočetní servery[1].

4.1 Volba implementačního prostředí

Pro implementaci byl zvoleno prostředí C++, paralelní zpracování pak realizováno jedním procesem o více vláknech. V dalších odstavcích následuje diskuse, ze které tato volba vzešla.

Objekty scény hrají v celé simulaci zásadní úlohu – emitují fotony, interagují s nimi a nakonec je pohlcují nebo zaznamenávají. Jak již bylo řečeno v části 3.3, všechny objekty scény lze rozdělit do několika skupin podle své úlohy ve scéně. Implementace každého takového vzorce chování je nezávislá na tvaru (geometrii) objektu, zřejmě je tedy při implementaci modelu scény vhodné vycházet z konceptu *objektově orientovaného programování*, jehož zásadním přínosem je právě abstrakce modelu chování od konkrétní instance objektu, což vede k jisté ortogonalizaci dílčích řešení a tím zpřehlednění a vyšší efektivitě kódu.

Omezíme-li se na objektově orientované programovací jazyky, dalším faktorem ovlivňujícím volbu prostředí je přenositelnost – zde se ukazuje jako nejvhodnější jazyk Java a C++, přičemž je-li kladen důraz na výpočetní výkon, jako optimální se jeví volba *programovacího jazyka C++*.

[1] přestože se v návrhu aplikace hovoří o výhodách paralelizaci úlohy, není cílem implementace rozhraní typu *MPI* (víceprocesorové a distribuované systémy)

Paralelní zpracování může být realizováno spuštěním více souběžných výpočetních procesů, nebo více vláken v rámci jednoho procesu. Obecně vzato, různé procesy běží v různých paměťových prostorech a je tedy nutné vyvinout jisté úsilí pro sdílení společných objektů, oproti tomu různá vlákna v rámci jednoho procesu se musí naopak vyvarovat neřízeného přístupu k prostředkům. Vzhledem k tomu, že jednotlivé podúlohy, jež se mají řešit paralelně, sdílejí velké množství dat (definice scény, fronta fotonů, konzolový výstup), je vhodnější druhé řešení, tj. vykonávat podúlohy *ve více vláknech*.

4.1.1 Externí knihovny

Jádro výpočtu je původní, nepoužívají se žádná existující řešení. Externí knihovny se používají pouze pro podporu paralelního zpracování a vstupně-výstupní operace se soubory ve standardních formátech.

Tabulka 4.1.1 uvádí přehled použitých knihoven a verzí, ve kterých jsou v aplikaci použité. Všechny knihovny jsou licencovány jako *open-source*[2] a jsou implementovány v jazyce C nebo C++, což zaručuje širokou přenositelnost mezi různými platformami.

knihovna	verze	jazyk	
`boost`	1.40.0	C++	třída `boost::thread` pro platformně nezávislou podporu řízení vláken
`expat`	2.0.1	C	čtení souborů ve formátu XML
`libpng`	1.2.39	C	čtení a zápis souborů ve formátu PNG
`zlib`	1.2.3	C	komprese dat, vnitřně použitá knihovnou `libpng`
`expreval`	3.5	C++	vyhodnocení aritmetických výrazů

4.2 Objektový model

Plná dokumentace objektového modelu je k dispozici na přiloženém CD, v tomto oddílu je model představen jen v hrubých rysech, popisem základních objektů.

V tabulce 4.1 je uveden přehled tříd, které představují rozhraní nejvýznamnějších objektů podílejících se na výpočtu.

4.2.1 Třída `ILight`

Společná všem zdrojům světla je schopnost emitovat fotony – třída `ILight` zveřejňuje jedinou metodu `EmitPhotons`, kterou se spustí emise fotonů. Vlastnosti světelného

[2]licence *open-source* umožňuje volné použití kódu, případná distribuce zdrojových kódů však musí obsahovat původní licenční podmínky

třída	použití
`ILight`	světelný zdroj
`IFigure`	těleso – optický prvek, stínítko, virtuální těleso
`ISheet`	obraz na stínítku
`IGeometry`	geometrie tělesa
`IMaterial`	vlastnosti materiálu tělesa
`IReflectionAlgorithm`	algoritmus pro zpracování fotonu
`ISheetRenderer`	algoritmus pro vykreslení obrazu stínítka do souboru

Tabulka 4.1: Přehled základních abstraktních tříd objektového modelu

zdroje jsou spravovány konkrétní instancí tohoto rozhraní, přístup ke scéně (především k frontě fotonů) je zajištěn referencí na strukturu `SceneContext` předanou v parametru této metody.

Všechny implementace tohot rozhraní musejí být *thread-safe* (bezpečné pro volání z více vláken najednou) – pro usnadnění vývoje (a eliminaci opakování kódu) byla vytvořena třída `MTLight` nabízející jistý „framework“ pro vícevláknové zdroje světla. Nové instance jej mohou, ale nemusejí využívat.

Významné implementace: `BitmapLight` pro bitmapový a `PointLight` pro bodový zdroj světla

4.2.2 Třída `IFigure`

Abstraktní těleso, jež může interagovat s fotony, příp. může být zobrazeno na *snímku kamery*. Tuto funkci zajišťují především metody

- `FindMeetingPoint` – hledá jeden nebo všechny průsečíky tělesa s daným fotonem
- `ProcessPhoton` – předá tělesu dopadající foton k zpracování
- `GetLuminance` – vrací barvu, jíž těleso v daném bodě a směru září (tuto metodu využívá *snímek kamery* pro stanovení barvy tělesa ve zkoumaném bodě)

Nejvýznamější implementací je třída `StdFigure` reprezentující běžné těleso popsané svým materiálem, geometrií, algoritmem zpracování fotonu, bitmapou a barvou pro *snímek kamery* – zde se významně těží z výhod objektově orientovaného návrhu, každou tuto vlastnost tělesa je možné snadno nastavit přiřazením vhodné instance. Tabulka 4.2 ukazuje konfiguraci geometrie a algoritmu zpracování fotonu pro některé typické objekty scény.

objekt scény	`IGeometry`	`IReflectionAlgorithm`
skleněná koule	`SphereGeometry`	`StdReflectionAlgorithm`
kruhová clona	`CSGGeometry`	`TrapReflectionAlgorithm`
zrcadlo	`RectGeometry`	`MirrorReflectionAlgorithm`
stínítko	`RectGeometry`	`RectSheetReflAlg`

Tabulka 4.2: Konfigurace objektu `StdSheet` pro typické objekty scény

4.2.3 Třída `ISheet`

Obecné pravoúhlé stínítko, umožňuje přístup k jednotlivým pixelu rastru, obsahuje veřejnou metodu `Render`, kterou některá implementace této třídy např. ukládá obsah stínítka do souboru.

4.2.4 Třída `IGeometry`

Geometrický model tělesa, umožňuje nalézt průsečík s polopřímkou (fotonem) a rozhodnout, zda daný bod je uvnitř tělesa nebo vně (využíváno CSG tělesy).

4.2.5 Třída `IMaterial`

Jediný materiál, který v této aplikaci neodráží ani nepohlcuje fotony, je bezbarvý průhledný materiál o daném indexu lomu. Třída `IMaterial` tedy popisuje pouze index lomu, jehož definice se liší v závislosti na tom, zda je požadována simulace *disperzního* chování, či nikoli.

Disperze ne Index lomu je kladné reálné číslo.

Disperze ano Index lomu závisí na vlnové délce, je proto vždy vypočten pro každý zpracovávaný foton. Příslušná závislost se definuje v definičním souboru scény třemi hodnotami indexu lomu pro tři různé vlnové délky, pro výpočet se používá vztah 2.1.

4.2.6 Třída `IReflectionAlgorithm`

Společné rozhraní pro algoritmus zpracování dopadajícího fotonu, přiřazením konkrétní implementace se konfiguruje těleso (pomocí struktury `FigureConfig`).

Třída obsahuje jedinou metodu, `ProcessPhoton`, které zpracuje dopadnutý foton – ten může být pohlcen, odražen, lomen, příp. rozdělen do více jiných fotonů.

4.2.7 Třída `ISheetRenderer`

Společné rozhraní pro algoritmy, které z obrazu na stíníku vytvářejí bitmapu, jediná metoda `Render` je volána po dokončení zpracování všech fotonů ve scéně.

4.3 Použité algoritmy a metody

4.3.1 Matematické výpočty

Jedním z cílů implementace je přesnost veškerých výpočtů, nejen že se nepoužívají žádná zjednodušení ani aproximace, všude, kde je to možné, eliminuje se také použití složitějších knihovních funkcí, které by mohly zavést do výpočtů chybu. Dalším faktorem, který ovlivňuje přesnost a zavádí jistá omezení, je datový typ zvolený pro uložení reálných čísel.

Reálná čísla Pro veškeré výpočty a uložení hodnot se reálná čísla modelují jako čísla v pohyblivé řádové čárce, pro aplikaci byl zvolen datový typ *double*.

Oblast nuly Protože uložení čísel v pohyblivé řádové čárce je zatíženo jistou chybou, nelze dvě taková čísla vzájemně porovnávat standardním predikátem rovnosti. Proto byla v aplikaci zavedena tzv. *oblast nuly*, která vymezuje maximální přesnost, která se při porovnání dvou čísel uvažuje; v aplikaci je stanovena konstantou `Zero` s hodnotou 10^{-6}. Je-li vzájemná vzdálenost dvou čísel menší než tato hodnota, jsou tato čísla považována za sobě rovná. To je třeba uvažovat při návrhu scény, tj. všechny rozměry a vzdálenosti by měly být větší než tato konstanta.

Operace Výpočty se omezují na základní operace sčítání, odčítání, násobení, dělení a druhou odmocninu. Goniometrické funkce se používají pouze v situacích, kde nemohou ovlivnit geometrii paprsku procházejícího scénou, např. při generování vektorů pro počáteční směr paprsku fotonu emitovaného ze zdroje.

4.3.2 Snímek kamery

Vykreslení snímku sestává ze dvou částí – objektů scény vykreslených metodou sledování paprsku (ray-tracing) a stop fotonů zachytávaných v průběhu výpočtu simulace.

Objekty scény Při sledování paprsku je volána pouze metoda `IFigure :: FindMeetingPoint`, ne však již následné `ProcessPhoton` – je tedy sledována pouze přímá viditelnost, paprsek není odrážen ani lomen; navíc scénu pouze zkoumá: dopadne-li na stínítko, nijak jej neovlivní. Barva tělesa je definována v materiálu tělesa (atribut `side-color`).

Stopy fotonů Vykreslování fotonů procházejících scénou je realizováno registrací události `IPhotonEvent`, která je volána v každém cyklu hlavní smyčky. Existuje-li průsečík fotonu se scénou, je v obsluze události zakreslena celá úsečka od počátku fotonu k průsečíku, v opačném případě se vykreslí úsečka do „nekonečna“, tj. do vzdálenosti definované v uzlu `<rendering>`.

4.4 Paralelní zpracování

Úzkým hrdlem celé koncepce jsou všechna stínítka, do kterých všechna výpočetní vlákna ukládají příspěvky energie, a *fronta fotonů*, kterou sdílí nejen výpočetní vlákna, ale i světelné zdroje.

Stínítka

Konfliktům při přístupu do paměti stínítek je možné se vyhnout – v implementaci se každé stínítko dělí do několika vrstev, počet vrstev je roven počtu výpočetních vláken. Každé vlákno má svůj identifikátor a když vyzvedne foton z fronty, uloží do jeho struktury svou identifikaci. Stínítko pak podle čísla vlákna, které je uloženo ve fotonu rozděluje příspěvky do vrstev. Tím jsou datově vlákna zcela oddělena a při přístupu do stínítek není nutná žádná synchronizace.

Fronta fotonů

Konfliktům ve *frontě fotonů* se v principu vyhnout nelze, jedná se o úložiště, kde si zdroje světla a výpočetní vlákna předávají fotony. Lze předejít alespoň části konfliktů, a to oddělením nových fotonů a fotonů, které se do fronty vracejí – v implementaci má každé vlákno vlastní frontu znovu zpracovávaných fotonů a není v přístupu k nim nijak limitováno.

Synchronizace vláken se tedy v celé aplikaci omezuje pouze na předávání nových fotonů světelnými zdroji výpočetním vláknům a na synchronizaci zápisu na standardní výstup (běhové informace, každou jednu sekundu dojde k výpisu jednoho řádku, viz 5.3).

4.5 Optimalizace

4.5.1 Rozšiřující vlastnosti průsečíku

Při zpracování fotonu na povrchu tělesa je zpravidla nutné znát normálový vektor v bodě dopadu fotonu. Je sice možné nechat těleso vypočítat normálu až v okamžiku zpracování fotonu tělesem, výhodnější je však normálu určit již při hledání průsečíku – zpravidla je zjištěna jako jeden z kroků výpočtu průsečíku. Proto je vhodné ukládat

normálu do struktury průsečíku `MeetingPoint`, přestože bude foton zpracován pouze pro jeden průsečík – nejbližší z nalezených.

4.5.2 Předpočítané vlastnosti fotonu

Některé vlastnosti objektu fotonu se ve scéně musejí vypočítat vícekrát, přitom vstupy pro výpočet jsou známy již v okamžiku vytvoření fotonu ve světelném zdroji. Nicméně, pokud by byl výpočet přesunut do světelného zdroje, mohl by se pro fotony, jež opustí scénu bez jediné kolize s objekty scény, tento výpočet provádět nadbytečně. Typickým příkladem je XYZ reprezentace barvy fotonu, kterou je možné vypočítat již v okamžiku vzniku fotonu, uplatní se ale pouze v případě, že foton dopadne na některé stínítko.

Proto byl implementován mechanismus odloženého výpočtu některých vlastností fotonu, kdy objekt fotonu obsahuje indikátor, zda byly tyto vlastnosti již napočteny, a referenci na algoritmus, jenž tyto vlastnosti na fotonu na požádání vypočte; k tomu dochází v okamžiku, kdy je nalezen průsečík fotonu se scénou nebo má být foton zobrazen na snímku kamery.

Kapitola 5

Uživatelská příručka

5.1 Instalace a spuštění programu

Na přiloženém CD se nachází instalační program pro Microsoft Windows 2000/XP, a to v adresáři *win32/setup*.

Program se spouští z příkazové řádky s následujícími parametry:

```
oLav.exe [-validate] [ [-var x=1] ...] scene.xml
```

Je-li uveden parametr *-validate*, je pouze ověřena validita definičního souboru včetně existence vstupních souborů, výpočet se nespouští.

Parametrem *-var* je možné definovat proměnnou, jejíž hodnotu lze použít v definičním souboru jako hodnotu atributu, proměnná může být číslo, text nebo prázdná hodnota.

5.2 Definice scény

Definiční soubor je soubor ve formátu XML s popisem scény a nastavením výpočtu. Přehled všech uzlů a atributů XML stromu je uveden v příloze A, v této kapitole je podrobněji popsán význam jednotlivých uzlů a hodnot atributů.

Definice scény:

```
<olav>
  <scene>
    <light-source/> ...   světelné zdroje
    <figure/> ...         definice těles
    <sheet/> ...          stínítka
  </scene>
  <configuration/>        nastavení programu
</olav>
```

5.2.1 Aritmetické výrazy

V definičním souboru je možné v hodnotách atributů použít aritmetické výrazy. Základní syntaxe použití proměnné je ${*výraz*}; je-li celá hodnota atributu tvořena výrazem, je možné vypustit složené závorky. Jsou povoleny např. následující zápisy:

```
atr="${x}"    atr="$-x"    atr="$x-1"    vector="1 ${y+1} $z-1"
```

Specifická situace nastává u hodnot vektorů – jednotlivé složky jsou odděleny mezerou, každá složka tedy může být číslo nebo samostatný výraz, žádný z výrazů navíc nesmí obsahovat mezery[1] (které jsou jinak ve výrazech povoleny). Následující výrazy tedy nejsou správné:

CHYBNĚ: `vector="${x + 1} 1 0"` `vector="${0 1 0}"`

Výraz může definovat novou proměnnou operací přiřazení (znak `=`), nelze však předefinovat již existující proměnnou. Ve výrazech je možné použít čísla, proměnné a základní aritmetické a goniometrické operace. POZOR: je-li ve výrazu použita proměnná, jež dosud nebyla definována, je automaticky definována a její hodnota je 0[2]. Výraz se může skládat z více částí oddělených středníkem, hodnotou celého výrazu je hodnota posledního podvýrazu. Následující příklad ukazuje použití vícenásobného výrazu a přiřazení – na základě parametru t je vypočten úhel *alfa*, souřadnice x, y a na výstup je vypsána hodnota úhlu *alfa*:

```
<variable name="alfa" value="$a=t/100*180;x=cos(a);y=sin(a);a"/>
```

Hodnotou výrazu může být pouze číslo, nejsou definovány žádné textové operace. Jediným způsobem, jak dynamicky sestavit textovou hodnotu, je použití uzlu `variable`:

```
<variable name="filename" value="img_${t}_${suffix}"/>
```

Výraz může definovat novou proměnnou operací přiřazení (znak `=`), každý výraz se může skládat z několika takových přiřazení oddělených středníkem; hodnota výrazu je rovna poslednímu přiřazení (podobně jako např. v programovacím jazyce C++).

5.2.2 Použití proměnných

Pro celý definiční soubor existuje jediná sada proměnných, hodnoty mohou být číselné nebo textové. Název proměnné se může skládat z písmen a číslic, nesmí však začínat číslicí. Pro oddělení více částí názvu lze použít tečku, např. `bod.x=1`.

[1] je to dáno postupem zpracování – nejprve jsou podle mezer nalezeny jednotlivé složky, pak teprve jsou vyhodnocovány případné výrazy

[2] toto chování vychází z použité knihovny *expreval*

Vyhodnocení proměnné Typ proměnné se nerozlišuje – je-li v místě použití proměnné očekáváno číslo, je hodnota považována za číslo, v případě neúspěšné konverze je hodnota výrazu rovna 0. Je-li očekávána textová hodnota, bere se původní hodnota, tak jak byla definována (např. s úvodními nulami).

Proměnné je možné definovat následujícím způsobem:

- V parametru příkazové řádky – proměnné jsou platné v celém definičním souboru.
- Ve výrazu – je-li ve výrazu přiřazena hodnota proměnné, stává se tato proměnná platnou pro zbytek definičního souboru.
- V uzlu `variable` – proměnná je platná pro zbytek souboru. Je-li uveden atribut `name`, je kromě přiřazení hodnota uvedena v logu aplikace (na standardním výstupu). Příklad použití:

  ```
  <variable name="alfa" value="$t/total*(alfa2-alfa1)+alfa1"/>
  ```

  ```
  <variable value="$a.x=len*cos(a); a.y=0; a.z=len*sin(a)"/>
  ```

  V prvním případě je vypočtena hodnota úhlu a objeví se ve výstupu aplikace, druhý zápis definuje tři nové proměnné bez informace na výstupu.

5.2.3 Světelný zdroj

Světlo vyzářené zdrojem je modelováno jako množina fotonů emitovaných do scény. Každý foton s sebou nese barevnou charakteristiku (spektrální složení), zlomek celkové energie zdroje a geometrii (polopřímka s počátkem na povrchu zdroje a směrovým vektorem daným geometrií zdroje).

Energie Celkový výkon světelného zdroje se nastavuje hodnotou zářivého toku (radiant-flux), nebo zářivou intenzitou (radiant-intensity), tj. tokem vztaženým na jednotku prostorového úhlu vyzařovaného světla. Výkon zdroje se projeví v počtu fotonů, které budou v simulaci emitovány ze zdroje – bude-li ve scéně jediný světelný zdroj, bez ohledu na jeho výkon z něj bude emitován celkový počet fotonů požadovaný v nastavení (viz 5.2.6), v případě více zdrojů se celkový počet fotonů rozdělí mezi zdroje v poměru jejich výkonů. Protože světelný účinek emitovaného světla může být různý (v závislosti na barvě), je možné nastavit korekci výkonu (power-correction), která nezmění počet fotonů zdroje, pouze nastaví hodnotu vnitřní energie každého emitovaného fotonu; to se projeví při výpočtu hodnoty barvy při dopadu fotonu na stínítko – fotony s vyšší výkonovou korekcí budou sytější. Takové nastavení je vhodné provést, pokud scéna obsahuje několik monochromatických zdrojů

různých barev a ve scéně má docházet k vzájemnému mísení jejich fotonů, viz scéna `scene02.xml` a odst. 7.1.2.

Geometrie Každý zdroj má svůj povrch, ze kterého jsou emitovány fotony, a množinu směrových vektorů, jichž fotony mohou nabývat. Bod na povrchu i směrový vektor jsou zvoleny náhodně s rovnoměrným rozdělením, rozsah hodnot směrových vektorů (rozptyl) je dán nastavením zdroje.

Světelný zdroj může být *bodový*, nebo *bitmapový*.

Bodový zdroj

Světlo generované bodovým zdrojem je homogenní, všechny fotony s sebou nesou shodnou barevnou informaci.

Definice bodového zdroje:

```
<light-source type="point">   bodový zdroj
  <emission/>                 geometrie
  <color type=".."/>          složení světla
</light-source>
```

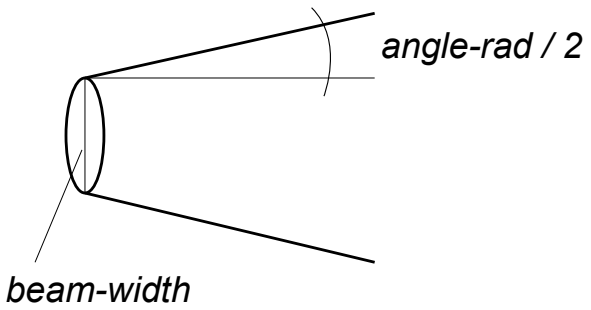
Poznámka: přehled všech atributů a jejich hodnot viz příloha A.

Geometrie Svazek paprsků může mít následující tvar (viz obr. 5.1):

1. válce – povrch zdroje vymezuje kružnice v rovině kolmé na vektor `vector` se středem v bodě `point` o průměru daném šířkou svazku `beam-width`

2. kuželu – počátek paprsku každého fotonu je daný bodem `point`, směr je dán vektorem `vector` vychýleným nejvýše o polovinu úhlu `angle-rad` v radiánech(resp. `angle-deg` ve stupních); udává se tedy úhel vymezující okraj svazku

3. komolého kuželu v případě, že je zároveň definována šířka svazku i úhel paprsků



Obrázek 5.1: Tvar paprsku bodového zdroje světla

Barva Složení světla je dáno typem světla (atribut `type`), je-li spektrální (`type = "spectrum"`), očekává se definice spektrální distribuce, v případě monochromatického zdroje světla (`type="monochromatic"`) je definicí vlnová délka, typ RGB (`type="rgb"`) převádí danou RGB barvu na spektrum (viz odst. 3.2.6), pro spektrální stínítka však není zaručeno, že bude zachován odstín barvy.

Při definici složení světla je možné využít přednastavené standardní zářiče definované komisí CIE: zářič A, D65 a F11.

Příklad:

```
<color use="cie-d65"/>  <!-- cie-d65, cie-a, cie-f11 -->
```

Bitmapový zdroj

Bitmapový zdroj generuje fotony různých barev v závislosti na hodnotách bitmapy namapované na obdélník umístěný ve scéně. Berou se v úvahu jen pixely, které mají větší jas než práh daný hodnotou atributu `treshold`

Směr fotonů je dán atributy `direction` – hodnota *normal* znamená směr normály k rovině bitmapy, hodnota *vector* očekává nastavení vektoru atributem `vector`. Pobobně jako u bodového zdroje je možné fotony ze směru vychýlit o úhel v rozmezí `angle-rad` (resp. `angle-deg`).

Definice bitmapového zdroje:

```
<light-source type="bitmap">   bitmapový zdroj
  <emission/>
  <rectangle/>                 materiál
  <file/>
</light-source>
```

5.2.4 Těleso

Těleso je popsánou svou geometrií (tvar a umístění tělesa ve scéně) a materiálem (optické vlastnosti tělesa). Ve scéně je možné definovat více těles, na pořadí záleží pouze v případě „slepování“ těles s jinými tělesy nebo stínítky.

Definice skutečného tělesa:

```
<figure>                          vlastnosti materiálu
  <sphere/> | <cylinder/> | ...   geometrie tělesa
  <material/>                     materiál
</figure>
```

Je-li těleso *skutečné*, je plnohodnotnou součástí scény – fotony interagují s jeho rozhraním, je vidět i na snímku kamery.

Definice virtuálního tělesa:

```
<figure layer="virtual">   vlastnosti materiálu
  <bitmap>
    <file/>                soubor s bitmapou (PNG, 24-bit RGB)
  <rectangle/>             geometrie bitmapy, pouze obdélník
</figure>
```

Je-li těleso *virtuální*, je součástí scény pouze pro vykreslení *snímku kamery*, nemá však vliv na výpočet šíření světla – je vhodné např. pro umístění libovolné bitmapy (měřítko, úhloměr) do scény pro usnadnění následné analýzy výstupního obrázku. Zavedení virtuálních těles tedy umožňuje přidávat do scény pomocné objekty, které výpočet nejen neovlivňují, ale také zbytečně nezpomalují. Soubor s bitmapou musí být ve formátu PNG s 24-bitovou barevnou hloubkou, cesta k souboru je relativní vzhledem k aktuálnímu adresáři (tj. ve kterém byl program spuštěn).

Geometrie

Pro popis tvaru tělesa se používají základní geometrické objekty – koule, poloprostor, válec a obdélník. Tyto objekty je možné kombinovat technikou CSG[3] množinových operací a vytvářet tak složitější tělesa.

Koule může být určena středem (`centre`) a poloměrem (`radius`), nebo svým pólem, vektorem ke středu a poloměrem

```
<sphere centre="20 0 0" radius="22"/>
<sphere pole="-2 0 0" vector="1 0 0" radius="22"/>
```

Poloprostor je vymezen hraniční rovinou danou normálovým vektorem (`normal`) procházející daným bodem; objem poloprostoru se nachází na straně roviny ve směru normály.

```
<halfspace point="1 0 0" normal="1 1 0"/>
```

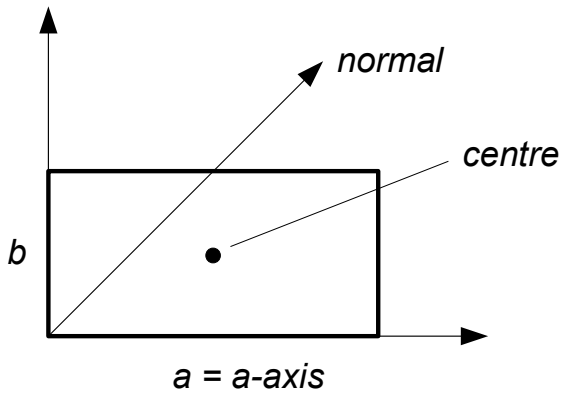
Válec je určen poloměrem (`radius`) a středem dolní (`base`) a horní podstavy (`cap`)

```
<cylinder base="0 0 0" cap="50 0 50" radius="5"/>
```

Obdélník se stranami a, b je popsán středem (`centre`), normálovým vektorem (`normal`), vektorem strany a a délkou stran a, b (`size` – vektor o délce 2); vektor strany b je vypočten tak, že $(a, normal, b)$ tvoří pravoúhlý, kladně orientovaný systém vektorů[4]

[3] Constructive Solid Geometry

[4] vnitřní souřadnice obdélníka vzrůstají podél vektorů stran a, b (to je důležité pro orientaci stínítka při jeho vykreslení a uložení do souboru)



Obrázek 5.2: Geometrie obdélníku

```
<rectangle
   centre="56 0 43" normal="-1 0 0" a-axis="0 1 0" size="26 26"/>
```

CSG těleso je definováno skupinou těles a množinovou operací, která je spojuje. Tělesa v množině jsou uložena jako poduzly uzlu `csg` a jsou definována svou geometrií libovolného typu, je tedy možné do sebe vnořit více úrovní CSG operací. Podporovány jsou operace průniku (`intersection`, těleso je průnikem všech těles v množině) a rozdílu (`difference`, od prvního tělesa se odečítají všechna ostatní)

```
<csg type="intersection"/>
```

Příklad (spojná čočka složená ze dvou kulových ploch):

```
<csg type="intersection">
    <sphere centre="20 0 0" radius="22"/>
    <sphere centre="-20 0 0" radius="22"/>
</csg>
```

Příklad (vyduté sférické zrcadlo):

```
<csg type="difference">
    <sphere centre="-80 0 0" radius="280.05"/>
    <sphere centre="-100 0 0" radius="300"/>
</csg>
```

Materiál

Materiál popisuje chování fotonu, který dopadl na povrch tělesa – zda se bude tento foton dále šířit scénou, v jakém směru a zda se případně změní některého jeho vlastnosti.

Definice:

`<material>`	vlastnosti materiálu
`<ior/>`	index lomu (má smysl jen pro průhledné materiály)
`</material>`	

Rozlišujeme následující typy materiálu:

1. průhledný (`translucent`) – po dopadu foton prochází do vnitřního objemu tělesa; očekává se definice poduzlu `<ior>` s definicí indexu lomu vnitřního prostředí. má-li index lomu různý od 1, dojde k lomu na rozhraní podle Snellova zákona lomu (po překročení mezního úhlu dochází k totálnímu odrazu, vlastnosti fotonu nejsou nijak upraveny). Index lomu vnějšího prostředí je vždy roven 1

2. zrcadlový (`mirror`) – foton se odráží od povrchu pod úhlem shodným s úhlem dopadu

3. optická past (`trap`) – foton je pohlcen a vyloučen z dalšího zpracování

Lom na rozhraní K lomu paprsku na rozhraní prostředí dochází na průhledných tělesech při vstupu fotonu do tělesa a při jeho opuštění. Na rozhraní je foton odchýlen od svého směru dle Snellova zákona lomu, jako vnitřní index lomu se použije index lomu materiálu tělesa, index lomu vnějšího prostředí je vždy roven 1. Dopadá-li foton pod větším než mezním úhlem, a tedy dochází k totálnímu odrazu na rozhraní, foton je zpracován jako by povrch tělesa byl zrcadlový.

Index lomu prostředí obecně závisí na vlnové délce procházejícího světla. Zda se při simulaci toto chování bude následovat, nebo se všechny paprsky budou lámat bez ohledu na vlnovou délku dopadajícího fotonu, je možné nastavit v uzlu materiálu hodnotou atributu `dispersion`. Při hodnotě `yes` bude index lomu záviset na vlnové délce a je tedy nutné tuto závislost popsat v poduzlu `ior` zadáním tří hodnot indexu lomu (`disp-value`) při libovolných vlnových délkách (`disp-lambda`); ostatní hodnoty budou spojitě dopočteny Cornu-ovým vzorcem 2.1. Vlnové délky je možné zadat číselně, nebo pomocí přednastavených hodnot Fraunhoferových čar písmeny A-H (viz 2.1.1). Nastavení materiálu `dispersion="no"` způsobí, že index lomu bude konstantní hodnota zadaná atributem `value` poduzlu `<ior>`.

Rozklad světla Je-li foton spektrální (složený z více vlnových délek), dochází při prvním průchodu průhledným tělesem k rozkladu světla do N fotonů o různých vlnových délkách, kde N je konstanta dána nastavením scény (viz 5.2.6). Přesněji, k rozkladu dochází při dopadu na povrch tělesa, které je průhledné a je pro něj požadována simulace disperze (nastavení `dispersion="yes"`). Původní foton je zrušen,

vlnová délka nových fotonů je zvolena náhodně s rozdělením definovaným ve světelném zdroji, ze kterého byl foton emitován, přičemž energie původního fotonu je rovnoměrně rozdělena mezi všech N nových fotonů – celková energie se tedy rozkladem nezmění. Protože každý nový foton má jedinou vlnovou délku, v souladu s teorií dále ve scéně již nedochází k dalšímu rozkladu světla.

Slepení těles Jsou-li při hledání průsečíku paprsku fotonu se scénou nalezena dvě nebo více těles, se kterými má foton stejný průsečík (tělesa se v bodě průsečíku „dotýkají“), je foton zpracován postupně všemi tělesy v pořadí, v jakém byly definovány v definičním souboru. Mají-li např. dvě takto slepená tělesa stejný index lomu, foton jejich rozhraním prochází bez změny směru (dojde k lomu na rozhraní těleso-okolí a následně okolí-těleso, při shodném indexu lomu se odchýlený paprsek vrací na druhém rozhraní do původního směru). Stále však platí, že tělesa se mohou pouze dotýkat; při průchodu překrývajícími se tělesy je chování fotonu nedefinované.

Pojmenování materiálu Pokud má být stejný materiál definován pro více těles, je možné při první definici v rámci definičního souboru materiál pojmenovat (atribut `name`) a místo definice materiálu pro jiné těleso tuto pojmenovanou definici použít (atribut `use`). Jsou-li v uzlu používajícím existující definici uvedeny další vlastnosti materiálu, jsou zohledněny, nový uzel materiálu může tedy pojmenovaný materiál použít pouze jako výchozí nastavení a některé vlastnosti pozměnit; provedené změny se přitom týkají pouze nového uzlu.

Vestavěné materiály V programu je několik předdefinovaných materiálů, které je možné použít při definici těles. Jedná se o průhledné materiály; nastaven je konstantní index lomu i tříhodnotová definice závislosti indexu lomu na vlnové délce. Jako výchozí je nastaveno `dispersion="yes"`, při použití materiálu je možné toto nastavení změnit. Jsou přednastaveny následující hodnoty indexu lomu (dle MFCh):

`<material>`			
`name`	`disp-value`	`disp-lambda`	
water	`1.331 1.333 1.337`	`C D F`	voda
glass	`1.457 1.459 1.464`	`C D F`	křemenné sklo
diamond	`2.4104 2.4175 2.4354`	`C D F`	diamant

Příklad použití:

```
<material use="glass"/>
<material use="glass" side-color="255 63 63"/>
```

5.2.5 Stínítko

Stínítko je rovina umístěná ve scéně, která ukládá informaci o dopadajícím světle a po ukončení výpočtu vykreslený obrazec ukládá do souboru jako rastrový obrázek.

Definice stínítka:

```
<sheet>
  <rectangle>    obdélník stínítka
  <file/>        soubor s výstupní bitmapou (PNG, 24-bit RGB)
  <rendering/>   konfigurace vykreslení
</figure>
```

Speciálním typem stínítka je *snímek kamery* (`type="sheet"`), který neukládá procházející světlo, ale je umístěn stranou od scény, zakresluje stopy fotonů procházejících scénou, výsledný obrazec je pak doplněn o zobrazení objektů scény (pro vykreslení se používá metoda *sledování paprsku*). Předpokládá se, že scéna je umístěna ve směru normály obdélníku.

Vykreslení je možné konfigurovat v uzlu `rendering`; je možné nastavit gamma-korekci (u *snímku kamery* se týká jen paprsků), u *snímku kamery* je možné zvolit, zda se mají vykreslovat jen fotony, které mají průsečík ve scéně, nebo scénu opouštějí (`photons="all"`), případně nastavit délku stopy fotonů opouštějících scénu (atribut `leaving-photon-length`).

5.2.6 Konfigurace výpočtu

Konfigurace:

```
<configuration>
  <photon>       počet fotonů, limit počtu iterací
  <threads/>     počet výpočetních i zdrojových vláken
</figure>
```

Nastavení fotonů

V sekci `photon` se nastavuje celkový počet fotonů, který bude v průběhu výpočtu emitován ze světelných zdrojů; rozdělení fotonů se řídí energií každého zdroje, viz 5.2.3.

Hodnotou `max-iterations` se nastavuje maximální počet interakcí fotonu se scénou, počítá se každý odraz a lom na rozhraní. Překročí-li foton tento limit, je zrušen.

Hodnota `disp-new-photons` udává počet fotonů, které budou vytvořeny v situaci, kdy se paprsek s definovaným spektrálním složením láme a vzniká spektrum. Nové fotony jsou monochromatické, jejich vlnová délka je generována náhodně s danou ditribucí (spektrální charakteristikou). Vyšší číslo znamená teoreticky větší efektivitu paralelizace (seznam nově vytvářených fotonů je privátní ve vláknu, které foton zpracovává, odpadá tak synchronizace s frontou fotonů).

Nastavení vláken

Udává se počet vláken světelných zdrojů (`light-threads`) a počet výpočetných vláken (`light-threads`). S vyšším počtem výpočetních vláken roste paměťová náročnost, neboť každé stínítko alokuje paměť pro ukládání příspěvku dopadajícího světla pro každé vlákno samostatně; opět odpadá nutnost synchronizace vláken.

5.3 Výstup programu

Po celou dobu běhu programu jsou na standardní výstup odesílány průběžné informace o činnosti programu.

Před zahájením výpočtu, během čtení definičního souboru a vytváření scény, jsou na výstup zapisovány informace o parametrech příkazové řádky, inicializaci objektů a nastavených proměnných.

V průběhu výpočtu se v pravidelných intervalech na výstup zapisují číselné hodnoty vnitřních čítačů rozlišené následujícími kódy:

`em` – počet celkem emitovaných fotonů

`pr` – celkem zpracovaných fotonů (počítá se každé hledání průsečíku fotonu se scénou)

`emQ` – velikost fronty fotonů

`ref` – počet odrazů a lomů fotonů

`noRef` – počet fotonů, které bez jediného průsečíku opouštějí scénu

`rThr` – počet běžících výpočetních vláken

Po ukončení výpočtu se zobrazuje počet zrušených fotonů (byl překročen maximální počet povolených odrazů / lomů), průběh ukládání obrázků stínítek a celkový čas běhu programu.

Kapitola 6

Příručka programátora

6.1 Kompilace

Pro sestavení kompilačního projektu se využívá utilita *cmake*, která nabízí jednotné rozhraní pro kompilaci na různých platformách. Definicí projektu je soubor *CMakeLists.txt*, který deklaruje používané knihovny, specifikuje cestu ke zdrojovým souborům a nastavuje další vlastnosti projektu. Kompilace s pomocí utility *cmake* sestává ze dvou kroků: vytvoření projektu a spuštení kompilace.

Veškeré soubory potřebné pro kompilaci aplikace jsou k dispozici na přiloženém CD, jejich uložení a postup kompilace se však významně liší pro zvolené prostředí.

6.1.1 Prostředí Unix

Jsou-li všechny knihovny instalovány ve standardním umístění, pro sestavení projektu stačí zadat následující příkazy

```
mkdir build
cmake ..
```

Pokud jsou některé knihovny instalovány na nestandardním umístění, je nutné pozměnit parametry:

```
cmake -D CMAKE_INCLUDE_PATH=<inc> -D CMAKE_LIBRARY_PATH=<lib> ..
```

kde se *<inc>* a *<lib>* nahradí cestou k hlavičkovým souborům a knihovně.

Spuštění kompilace se následně provede standardním příkazem `make`.

6.1.2 Prostředí Windows

Pro kompilaci v prostředí Windows je určen adresář *win32* na přiloženém CD, obsahuje již přeložené knihovny (adresář *lib*) i připravený projekt pro kompilaci aplikace (adresář *build*) určený pro *Visual Studio 8 2005*.

Využití připraveného projektu Projekt je provázán do adresáře s knihovnami, obsahuje však absolutní cesty, které předpokládají, že obsah CD je na disku *Z:*. Pokud tohoto stavu nelze dosáhnout (např. nasdílením a mapováním jakoby síťového disku), je možné v projektu cestu ručně změnit, soubory projektu jsou ve formátu XML.

Sestavení vlastního projektu Využívá se utilita *cmake*, například tímto postupem (předpokládá se vymazání obsahu adresáře *win32/build*)

```
cd win32\build
cmake ..\..
```

Vytvoří se projekt pro defaultní kompiler systému, příp. lze vnutit jiný (viz dokumentace k *cmake*, např. příkaz `cmake --help`)

Rekompilace knihoven V případě potřeby jsou v adresáři *lib-src* umístěny projekty pro kompilaci knihoven, viz informace v souborech *readme.txt* a *boost/readme.txt*.

6.2 Základní orientace v kódu

Kód se dělí do tří jmenných prostorů: *oLav* obsahuje výpočetní část aplikace, *xml* zahrnuje proces čtení definičního souboru a sestavení scény a *Const* je prostor pro konstanty, např. specifikaci spektrální distribuce.

Sestavení scény Datové struktury scény se vytvářejí při čtení definičního souboru, které zajišťuje třída *XMLReader* s využitím objektů „šitých na míru“ konkrétním uzlům XML stromu definovaným v souboru *XMLReaderType.cpp*

Hlavní smyčka Hlavní smyčku výpočtu (viz odst. 3.1.1) realizuje funkce *ProcessPhotons* v souboru *Scene.cpp* spouštěná v samostatných vláknech.

Výchozí hodnoty atributů Výchozí hodnoty atributů XML definice scény jsou zpravidla v konstruktorech datových typů pro uzly v souboru *XMLReaderType.cpp*, složitější pravidla a inicializace se provádějí buď v metodách týchž objektů, nebo přímo při čtení definice ve třídě *XMLReader*.

Interakce fotonu s rozhraním Po nalezení průsečíku fotonu s rozhraním je foton předá příslušnému algoritmu pro zpracování fotonu na rozhraní. Tímto algoritmem je zpravidla třída *StdReflectionAlgorithm*, která podle vlastností fotonu, průsečíku a materiálu rozhodne o dalším osudu fotonu – předává foton do fronty nebo jinému tělesu, příp. vytváří nové fotony vznikajícího spektra.

6.3 Synchronizace vláken

Vlákna jsou rozdělena do dvou skupin, v jedné jsou vlákna světelných zdrojů, druhou skupinu tvoří vlákna výpočetní. Hlavním úkolem synchronizace v obou skupinách je řízení přístupu k frontě fotonů.

Světelné zdroje Vlákna jsou přidělována jednotlivým světelným zdrojům v třídě *GroupLight*, práce s vlákny je pro oba typy zdrojů *PointLight* i *BitmapLight* soustředěna do jejich společného předka *MTLight*.

Výpočetní vlákna Veškerá synchronizace výpočetních vláken se nachází v *hlavní smyčce* (viz výše), zde dochází k jediné synchronizaci, další procesy (generování náhodného čísla, ukládání příspěvku do stínítka) již synchronizaci nevyžadují – stínítka jsou rozdělena na vrstvy, každé vlákno ukládá příspěvek do jiné vrstvy. to je realizováno očíslováním vláken, číslo vlákna je uloženo do každého nově vyzvednutého fotonu.

Kapitola 7

Výsledky

7.1 Správnost výpočtu

Při vývoji aplikace byl kladen důraz především na správnost výpočtu, v této části jsou popsány úlohy, na kterých byla správnost ověřena.

7.1.1 Spojná čočka

V této úloze se ověřuje geometrická správnost zobrazení spojnou čočkou. Scéna bude sestavena ze zdroje světla, bikonvexní čočky a stínítka v obrazové rovině. Poloha stínítka bude stanovena výpočtem, je-li simulace správná, bude na stínítku vykreslen ostrý obraz předmětu před optickým prvkem. Pro ověření správnosti sestavení scény je ve scéně umístěno měřítko, které bude zobrazeno spolu s optickou soustavou pouze na *snímku kamery*.

Pro ověření korektního chování i v případě spektrálního zdroje fotonů, bude zdroj emitovat fotony o různých vlnových délkách; protože je index lomu pro různé vlnové délky zpravidla různý, bude ve scéně umístěno více stínítek pro jednotlivé vlnové délky.

Analytický popis zobrazení na kulové ploše se zpravidla omezuje na paraxiální oblast, svazek paprsků je tedy omezen kruhovou clonou umístěnou před čočkou.

Úloha testuje základní geometrické výpočty – nalezení průsečíku s kulovou plochou, CSG operaci průniku, aplikaci Snellova zákona lomu, průmět skutečných i virtuálních optických prvků.

Konfigurace scény Čočka o tloušťce $4mm$ je tvořena dvěma kulovými plochami o poloměru $22\,mm$ a je centrována v počátku souřadnic. Zdrojem světla je bitmapa ve vzdálenosti $148mm$ od čočky generující fotony o vlnových délkách 700, 546,1 a $460\,nm$[1]. Materiál čočky je křemenné sklo s indexem lomu

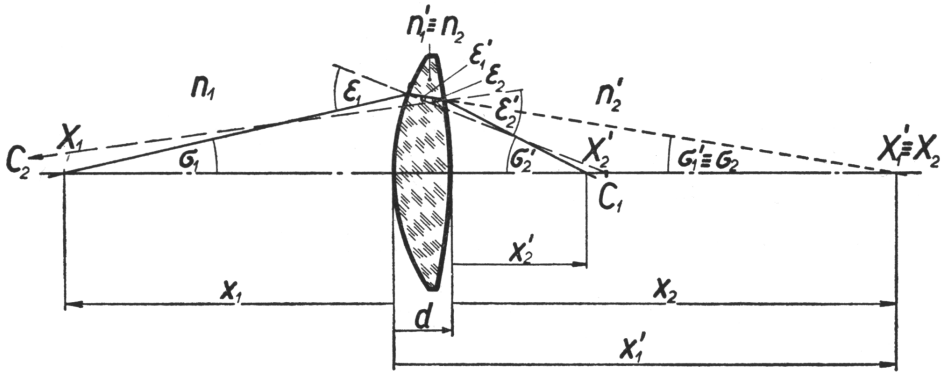
[1] dané výchozí hodnotou atribute `rgb-lambda` uzlu `emission`

$$n_1 = 1{,}457 \quad pro \quad \lambda_1 = 656{,}28\,nm,$$
$$n_2 = 1{,}459 \quad pro \quad \lambda_2 = 589{,}3\,nm,$$
$$n_3 = 1{,}464 \quad pro \quad \lambda_3 = 486{,}14\,nm.$$

Výpočet Z výše uvedené definice indexu lomu nelze přímo určit index lomu pro dané vlnové délky emitovaných fotonů. Je však možné – ve shodě s aplikací – použít *Cornu*-ův vzorec (viz), dostáváme pak pro vlnové délky fotonů indexy lomu N_i po řadě 1,455991, 1,460717 a 1,465931 nm.

Do vztahů uvedených v [2] pro výpočet obrazové roviny (viz obr. 7.1),

$$x_1' = \frac{n_1'}{\frac{n_1}{x_1} + \frac{n_1' - n_1}{r_1}}; \quad x_2 = x_1' - d$$
$$x_2' = \frac{n_2'}{\frac{n_2}{x_2} + \frac{n_2' - n_2}{r_2}}$$



Obrázek 7.1: K výpočtu polohy obrazu

dosazením

$$x_1 = -148, n_1 = n_2' = 1, n_1' = n_2 = N_i, r_1 = 22, r_2 = -22, d = 4$$

vychází vzdálenost předmětové roviny od čočky x_2' po řadě 28,3651, 28,0113 a 27,63088[2].

Výstup simulace odpovídá očekávání (viz obr. C.1), každé stínítko se nachází právě v jedné obrazové rovině, obraz je ostrý pouze pro jednu barevnou složku.

[2]Pro získání absolutních souřadnic je nutné tuto vzdálenost posunout o polohu druhé plochy čočky, tj přičíst 2 mm.

7.1.2 Disperze

Pro ověření disperzních vlastností byl zvolen lom světla na rozhraní dvou prostředí o různém indexu lomu. Zdroj světla je bodový s úzkým rovnoběžným paprskem světla, paprsek prochází z prostředí opticky hustšího do prostředí opticky řidšího, může tak nastat i totální odraz. Spektrální složení světla odpovídá světlu dennímu (dle specifikace CIE-D65).

Úloha ověřuje aplikaci Snellova zákona lomu, detekci totálního odrazu, rozklad světla do spektra.

Výpočet Pro ověření chování v oblasti blízké totálnímu odrazu by bylo možné definovat několik paprsků v úhlech blízkých meznímu úhlu. Stejnou vypovídací schopnost nicméně bude mít jiný případ: jediný, plnospektrální paprsek. Pokud se úhel dopadu zvolí tak, že je mezním pro vlnovou délku přibližně v „polovině“ spektra, část spektra s delší vlnovou délkou (červenější) se bude lámat za vzniku spektra, krátkovlnná část paprsku se bude odrážet zpět do prostředí jako nerozložený paprsek přibližně modré barvy.

Pro usnadnění výpočtu (bez újmy na obecnosti) volme jako hraniční vlnovou délku *Fraunhoferovu čáru D* (589,3 nm), pro ni máme ve výše uvedené definici daný index lomu 1,459. Pro výpočet mezního úhlu dosadíme do Snellova zákona lomu:

$$n_1 \sin\alpha = n_2 \sin\beta, \quad \sin\beta = 1$$
$$\sin\alpha = \frac{n_2}{n_1}$$

Je-li $n_1 = 1{,}459, n_2 = 1$, pak $\alpha = 43^\circ\, 16'\, 1{,}73''$.

Obr. C.3 potvrzuje, že se simulace chová očekávaným způsobem.

7.1.3 CSG těleso

Pro demonstraci konstruování těles metodou CSG byl vytvořen netriviální prvek sestávající z násobné CSG definice o třech úrovních vnoření – ze skleněné krychle jsou vyříznuty tři válce v osách x, y, z, rozdíl pak vstupuje do operace průniku spolu s koulí ořezávající vrcholy krychle. Uvnitř tělesa jsou dva monochromatické zdroje světla, výstupem je pohled na scénu, viz obr. C.4.

7.1.4 Skládání barev

Ve scéně jsou umístěny tři monochromatické zdroje o vlnových délkách 700, 546,1 a 460 nm[3] Jsou směrovány na stejné stínítko, na kterém je možné sledovat mísení barev řady kombinací.

[3] tyto vlnové délky nejvíce odpovídají červené, modré, resp. zelené barvě.

Aby výsledek součtu všech tří základních barev byla bílá barva, je třeba zdroje světla kalibrovat, tj. nastavit intenzity jednotlivých zdrojů světla – definuje se tím vlastně spektrální rozdělení. Pro výše uvedené vlnové délky byly experimentálně stanoveny intenzity po řadě 62,48, 1,66 a 1.

Výstup simulace odpovídá intuitivní představě, viz obr. C.5.

7.1.5 Bitmapový zdroj

Jako bitmapa zdroje světla byla zvolena fotografie, zobrazení bylo provedeno Taylorovým tripletem[4], přičemž vlnové délky pro složky r, g, b a jejich intenzity jsou nastaveny jako v předchozím odstavci[5].

Výstupní obrázek C.6 je neostrý, což je pravděpodobně způsobeno nepřesnou konstrukcí objektivu, další vadou je zřetelný barevný posun ve srovnání se vstupní bitmapou. Domnívám se, že se jedná o chybu metody, jednoduchý rozklad podle složek r, g, b popsaný v odst. 3.2.6 je zřejmě nedostatečný.

7.1.6 Lepené prvky

Následující úloha demonstruje možnost umístit dva optické prvky těsně k sobě a simulovat tak „slepení“ dvou těles, v praxi používané např. při konstrukci objektivů. Mají-li lepené prvky shodný index lomu, nedochází na rozhraní k lomu, a tedy ani k rozkladu světla na spektrum.

Scéna pro tuto úlohu se skládá z několika slepených prvků o stejném indexu lomu a jednoho prvku s jiným indexem lomu. Zdrojem světla jsou dva paprsky, jeden vstupuje do tělesa kolmo k povrhu, prochází několika rozhraními materiálů o stejném indexu lomu, těleso opouští opět kolmo k povrchu, nedochází tedy k rozkladu na spektrum. Druhý paprsek na jednom rozhraní prochází mezi dvěma materiály o různém indexu lomu, dochází k lomu a paprsek se rozkládá na spektrum, viz obr. C.5.

7.2 Efektivita paralelizace

Byla provedena měření na následujících strojích:

nb: Intel Core 2 Duo T7300 @ 2 GHz (32 bit, 2 jádra, 2 vlákna)

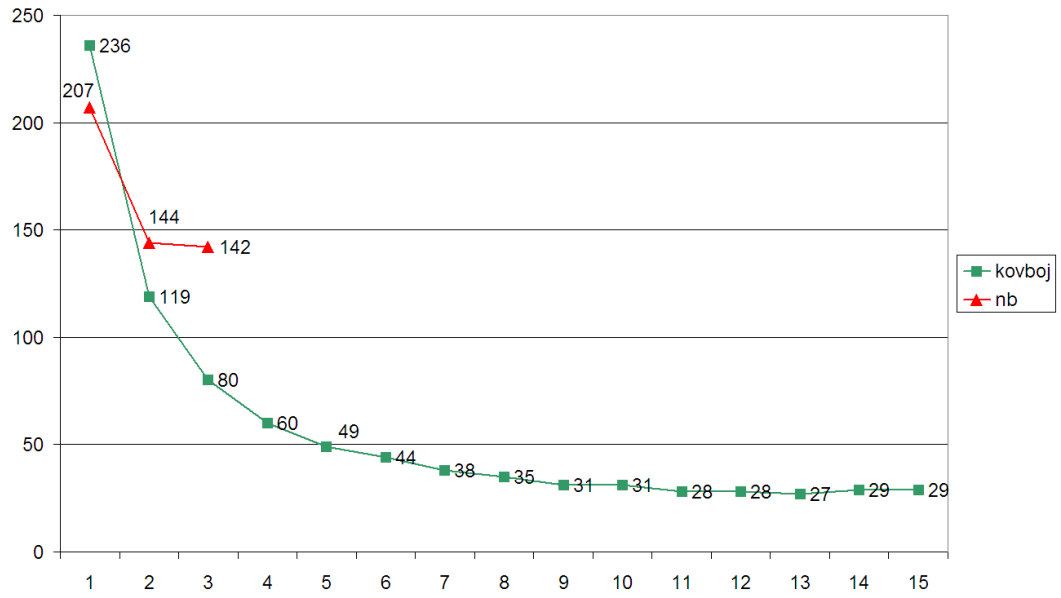
kovboj: 2x AMD Opteron 2431 @ 2.4 GHz (64 bit, 2× 6 jader, 12 vláken)

[4]konstrukce objektivu volně dle schematu v [2]

[5]tyto hodnoty nejsou v definičním souboru scény uvedeny, byly totiž implementovány jako výchozí

Byla zvolena testovací scéna *scene07.xml* (*Jednoduchost spektrálních barev*, viz obr. C.9) s 20 000 fotony a 100 novými fotony na rozhraní[6] a tři testy:

1. Ze scény jsou odstraněna všechna stínítka, paprsky procházejí přes všechny optické prvky jako ve výchozí scéně
2. Je odstraněn pouze *snímek kamery*, stínítko je zachováno
3. Je odstraněno pouze stínítko, *snímek kamery* je zachován



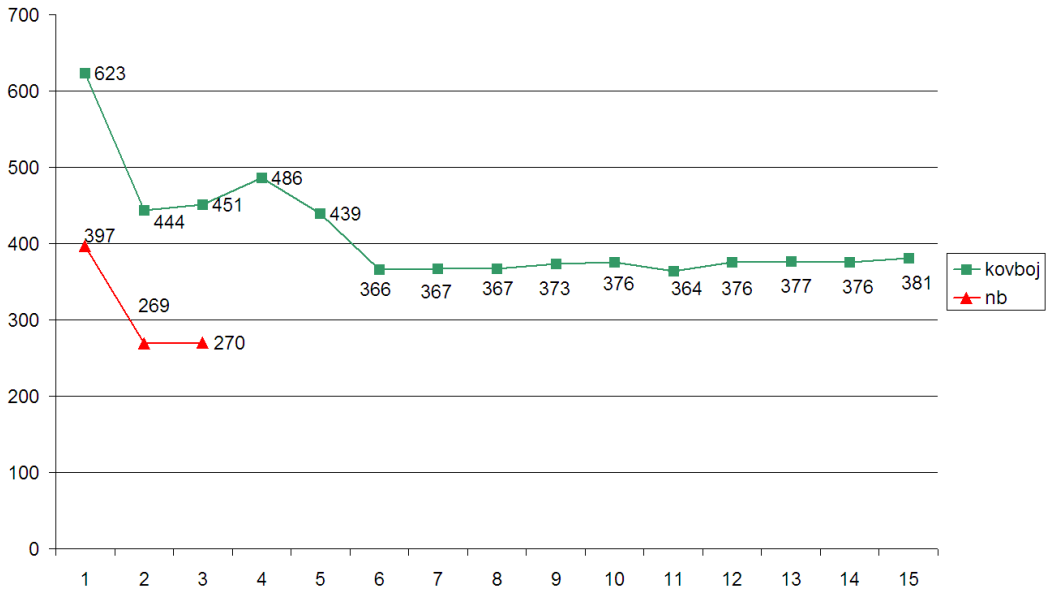
Obrázek 7.2: Délka výpočtu v závislosti na počtu vláken, test č. 1

Na výsledcích prvního testu (viz obr. 7.2) se ukazuje zrychlení cca 8,7× na 12vláknovém stroji a 1,4× na dvouvláknovém. Efektivita výpočtu není špatná, mohla by však být lepší.

Druhý test je poněkud překvapující (viz obr. 7.3) – přítomnost *snímku kamery* srazila efektivitu paralelizace na minimum, ještě více zarážející je však výkon silnějšího stroje, který je na jednom vlákně o polovinu pomalejší než stroj **nb**, dokonce ani na dvanácti vláknech nepředčí jeho výkon na dvou vláknech. Celkové zrychlení z jednoho na dvanáct vláken činí asi 1,6.

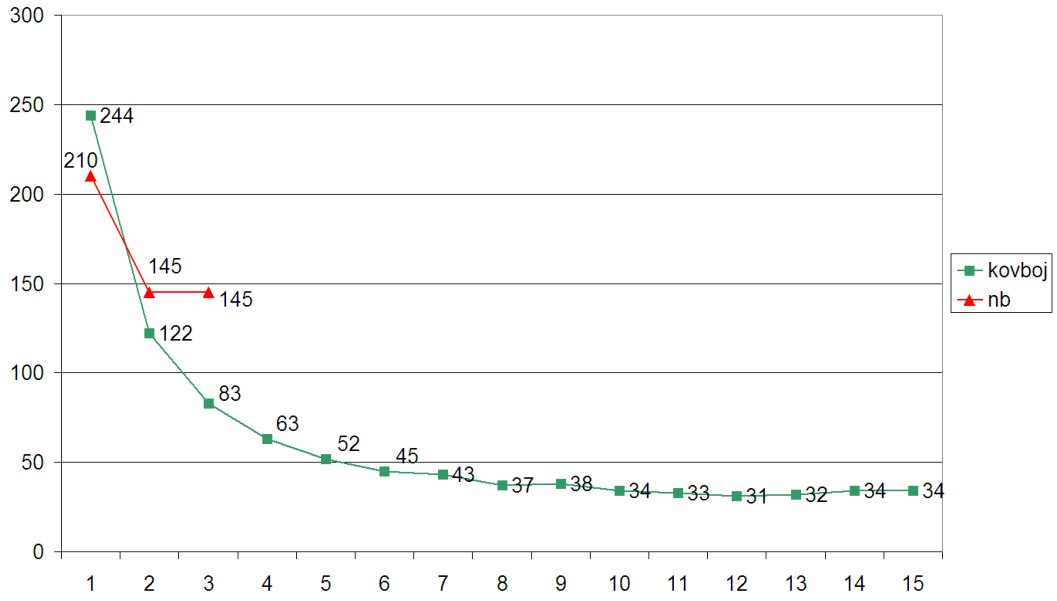
V třetím testu (viz obr. 7.4) se potvrzuje, že za nečekaným propadem výkonu stál skutečně *snímek kamery* – přítomnost stínítka téměř efektivitu ani absolutní výkon neovlivnila, výsledky jsou téměř totožné.

[6]nové fotony na rozhraní – počet nových fotonů, které jsou vytvořeny na prvním rozhraní, je-li paprsek spektrální; celkem je tedy ve scéně 20 000 × 100 fotonů, tj. 2 mil. fotonů



Obrázek 7.3: Délka výpočtu v závislosti na počtu vláken, test č. 2

Ve výpočtu *snímku kamery* nedochází k žádným konfliktům mezi vlákny, každé vlákno vykresluje stopy fotonů do jiné vrstvy, až po dokončení výpočtu se tyto vrstvy slévají do jedné. Od stínítka se výpočet liší jen výrazně vyšší koncentrací numerických výpočtů v jednom okamžiku, a to během kreslení úsečky stopy fotonu. Objasnění pravé příčiny by mohlo být předmětem dalšího výzkumu.



Obrázek 7.4: Délka výpočtu v závislosti na počtu vláken, test č. 3

Kapitola 8

Závěr

Bakalářská práce se zabývá návrhem a implementací systému pro simulaci spektrálních optických jevů. Vznikl systém, který splňuje zadané cíle a je použitelný jako didaktická pomůcka při výuce optiky i pro poloprofesionální využití při návrhu optické soustavy, příp. jiných specifických experimentech.

Práce vychází z teoretických poznatků o chování světla v optickém prostředí a lomu na rozhraní, na jejich základě definuje světelný model, který pokrývá veškeré chování světla požadované v simulaci.

Implementací v programovacím jazyce C++ byla vytvořena aplikace *oLav*. Rozhraní aplikace je konzolové, a tedy nezávislé na grafickém prostředí, aplikace je plně přenositelná[1] a podporuje paralelní zpracování na vícevláknových procesorech.

Významnou schopností programu je možnost používat v definičním souboru scény aritmetické výrazy a proměnné z příkazové řádky. Je tak možné spouštět dávkové úlohy pro generování sekvencí obrázků bez nutnosti pro každý snímek vytvářet samostatný definiční soubor – velmi se tak usnadňuje příprava animací založených na změně vlastností některých objektů scény.

Praktické výsledky jsou shrnuty v kap. 7, na přiloženém CD jsou k dispozici kromě obrázků v digitální podobě i ukázkové animace sestavené ze sekvencí scén. Tyto a další vypočtené materiály jsou také k dispozici na webové adrese

`http://cgg.mff.cuni.cz/thesis/kavalir/`

8.1 Další vývoj

Návrh aplikace je otevřený pro řadu rozšíření, které mohou vzejít ze specifických požadavků při řešení praktických úloh.

[1] aplikace byla testována na systémech Windows XP 32bit, Debian Linux 32/64bit a Windows 2008 Server 64bit

Virtuální text Kromě virtuálních objektů (měřítka, úhloměry uvažované pouze pro vykreslení *snímku kamery*) je možné zavést i vykreslování znaků. Obraz by tak mohl být doplněn o informace vztahující se ke scéně – poloha stínítka, úhel dopadu paprsku nebo světlost clony. Zvláště při vytváření animací může číselný údaj přispět k názornosti experimentu.

Interference Při zpracování fotonů je uvažována především geometrie (počátek a směr) a vlnová délka fotonu. Pokud by byly vlastnosti fotonu rozšířeny o fázový posun, je možné na každém průsečíku fotonu vyhodnotit fázi, s kterou foton dopadá na povrch a odpovídajícím způsobem upravit intenzitu příspěvku do obrazu stínítka. Spolu s definicí koherentního zdroje světla by tak v simulaci mohly být sledován interferenční jev.

Energetická bilance Při praktické konstrukci optické soustavy může být důležité uvažovat celkovou energii fokusovaného svazku světla v určitém bodě – při jeho překročení může např. dojít k poškození čočky. Současná implementace sleduje celkový světelný tok dopadající na stínítko, může být však rozšířena o přesnější specifikaci sledované oblasti.

Grafická nadstavba V průběhu výpočtu jsou na standardní výstup generovány stavové informace, je tedy možné vyvinout grafickou aplikaci, která bude simulaci spouštět, stavové informace zpracovávat a např. graficky zobrazovat. V kombinaci s 3D editorem by se pak mohla simulace stát kompaktním nástrojem pro spektrální simulace.

Literatura

[1] P. Dutré, K. Bala, P. Bekaert: *Advanced Global Illumination*,

[2] J. Fuka, B. Havelka: *Optika a atomová fyzika - I Optika*,

[3] C. Kolb, D. Mitchell, P. Hanrahan: *A Realistic Camera Model for Computer Graphics*, proceedings of SIGGRAPH 95 (Los Angelos, CA, August 6-11, 1995)

[4] E. Lafortune: *Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering*, PhD dissertation, Katholieke Universiteit Leuven, 1996

[5] `http://en.wikipedia.org/wiki/CIE_1931_color_space`, dne 25.5.2010

Příloha A

Přehled uzlů definičního souboru

V tomto seznamu jsou uvedeny všechny uzly, které se mohou vyskytnou v definičním souboru scény. U každého uzlu jsou popsány atributy a hodnoty, jichž mohou nabývat, dále pak poloha uzlu v rámci stromu dokumentu – jsou uvedeny uzly, které mohou být rodičovské a uzly, které mohou být potomky.

Pro zjednodušení zápisu je zaveden znak `@` pro označení atributu, zápis

- `@type` znamená „atribut `type`“,
- `color/@type` pak „atribut `type` uzlu `color`“ a
- `../@type` označuje „atribut `type` rodičovského uzlu“.

Hodnoty popisující vlastnosti objektů se definují v atributech příslušných uzlů XML stromu, pro datové typy hodnot je v dalším textu zavedeno následující značení:

- **text**: libovolný text neobsahující znaky `"` (uvozovka) a `>` (pravá hranatá závorka)
- **num**: desetinná místa se oddělují desetinnou tečkou
- **num**N: posloupnost N čísel oddělených mezerou (např. `point="1.5 -1 0"`)
- **RGB**: hodnoty barevných složek R, G, B z rozsahu $< 0, 255 >$
- **bool**: jako `nepravda` se chápe hodnota `"no"`, `"0"`, nebo prázdná hodnota `""`; ostatní hodnoty se vyhodnocují jako `pravda`
- **enum**: výčtové hodnoty (seznam platných hodnot vždy uveden v poznámce)

`<bitmap>`

Je potomkem uzlu: `figure`
Je rodičem uzlu: `file`

Textura mapovaná na povrch tělesa (implementováno jen těleso `<rectangle>`.

Nemá atributy.

`<color>`

Je potomkem uzlu: `light-source`
Je rodičem uzlu: `-`

Barevné vlastnosti fotonů emitovaných ze zdroje. Atributy lze rozdělit do tří skupin:

- společné

atribut	*typ*	*výchozí*	*hodnoty*
`type`	enum	–	`rgb, monochromatic, spectrum`
`rgb-color`	RGB	*white*	barva RGB
`lambda`	num	*$D*	vlnová délka

- specifické pro typ *rgb*

atribut	*typ*	*výchozí*	*hodnoty*
`rgb-lambda`	num3	700; 546,1; 460	vlnová délka složek
`rgb-power`	num3	62,48; 1,127; 1	intenzita složek
`threshold`	num	0,1	minimální jas pixelu v bitmapě
`spectral`	num	*yes*	zda je spektrální

- specifické pro typ *spectrum* zdroj

atribut	*typ*	*výchozí*	*hodnoty*
`lambda-min`	num		počátek intervalu
`lambda-max`	num		a konec intervalu
`lambda-step`	num		krok vlnové délky
`intensity`	num$N$		pole intenzit vlnových délek

`<configuration>`

Je potomkem uzlu: `olav`
Je rodičem uzlu: `photon, threads`

Nastavení výpočtu – počet fotonů celkem emitovaných do scény, počet výpočetních vláken.

Nemá atributy.

`<csg>`

Je potomkem uzlu: `figure, csg`
Je rodičem uzlu: `rectangle, halfspace, sphere, cylinder, csg`

CSG těleso

atribut	*typ*	*výchozí*	*hodnoty*
`type`	enum	–	typ operace – `difference, intersection`

`<cylinder>`

Je potomkem uzlu: `figure, csg`
Je rodičem uzlu: `-`

Geometrie válcového tělesa

atribut	*typ*	*výchozí*	*hodnoty*
`base`	num3	–	střed spodní podstavy
`cap`	num3	–	střed horní podstavy
`radius`	num	–	poloměr podstavy

`<emission>`

Je potomkem uzlu: `light-source`
Je rodičem uzlu: `-`

Geometrické a energetické vlastnosti fotonů emitovaných ze zdroje.

atribut	*typ*	*výchozí*	*hodnoty*
`direction`	enum	–	`vector, normal, hemisphere, geometry, sphere`
`point`	num3	–	počátek fotonů
`vector`	num3	–	směrový vektor fotonů
`color`	RGB	*white*	výchozí barva (nahrazuje uzel <color>)
`angle-rad`	num	0	rozptyl fotonů, $\langle 0, \pi)$
`angle-deg`	num	0	rozptyl fotonů, $\langle 0, 90)$
`beam-width`	num	0	šířka svazku
`power-correction`	num	1	korekce energie
`radiant-flux`	num	1	světelný tok (počet fotonů)
`radiant-intensity`	num	–	světelná intenzita

`<figure>`

Je potomkem uzlu: `scene`
Je rodičem uzlu: `geometry, material`

Těleso, optický prvek – je dán svou geometrií a materiálem povrchu, resp. objemu

atribut	*typ*	*výchozí*	*hodnoty*
`layer`	enum	*real*	`virtual, real`

`<file>`

Je potomkem uzlu: `light-source, bitmap, sheet`
Je rodičem uzlu: `-`

Název souboru (relativní k aktuálnímu adresáři) pro bitmapu zdroje, bitmapu virtuálního tělesa, resp. název souboru a rozlišení výstupní bitmapy stínítka

atribut	*typ*	*výchozí*	*hodnoty*
`file`	string	–	název souboru
`resolution`	num2	–	rozlišení bitmapy stínítka

`<halfspace>`

Je potomkem uzlu: `figure, csg`
Je rodičem uzlu: `-`

Geometrie – poloprostor daný bodem a normálovým vektorem (objem poloprostou je orientován ve směru normály).

atribut	*typ*	*výchozí*	*hodnoty*
`point`	num3	–	bod na hranici poloprostoru
`normal`	num3	–	normálový vektor

`<ior>`

Je potomkem uzlu: `material`
Je rodičem uzlu: `-`

Index lomu prostředí, definice je různá pro disperzní a nedisperzní materiály (atribut `material/@dispersion`):

- disperzní materiály – jsou očekávány tři hodnoty indexu lomu pro tři různé vlnové délky:

atribut	*typ*	*výchozí*	*hodnoty*
`disp-ior`	num3		tři hodnoty indexu lomu
`disp-lambda`	num3	*$C $D $F*	tři vlnové délky

- nedisperzní materiály – všechny paprsky se lámou s jediným indexem lomu

atribut	*typ*	*výchozí*	*hodnoty*
`value`	num	–	hodnota indexu lomu

`<light-source>`

Je potomkem uzlu: `scene`

Je rodičem uzlu: `emission, rectangle, file, color`

Světelný zdroj

atribut	*typ*	*výchozí*	*hodnoty*
`type`	enum	–	`bitmap, point`

`<material>`

Je potomkem uzlu: `figure`

Je rodičem uzlu: `ior`

Materiál popisuje optické vlastnosti tělesa

atribut	*typ*	*výchozí*	*hodnoty*
`type`	enum	–	`mirror, translucent, trap`
`side-color`	RGB	(63, 63, 63)	
`dispersion`	bool	yes	

`<olav>`

Je potomkem uzlu: `-`

Je rodičem uzlu: `scene`

configuration Kořenový uzel

Nemá atributy.

`<photon>`

Je potomkem uzlu: `-`

Je rodičem uzlu: `configuration`

Nastavení fotonu

atribut	*typ*	*výchozí*	*hodnoty*
`count`	num	–	celkový počet fotonů
`max-iterations`	num	10	maximální počet lomů / odrazů
`disp-new-photons`	num	1	počet nových fotonů při rozkladu

`<rectangle>`

Je potomkem uzlu: `csg, light-source, sheet`

Je rodičem uzlu: `-`

Geometrie obdélníku

atribut	*typ*	*výchozí*	*hodnoty*
`centre`	num3	–	střed obdélníku
`normal`	num3	–	normálový vektor roviny
`a-axis`	num3	–	osa strany b
`size`	num2	–	rozměr, délka strany a a b

`<rendering>`

Je potomkem uzlu: `sheet`
Je rodičem uzlu: `-`

Nastavení vykreslení

atribut	*typ*	*výchozí*	*hodnoty*
`photons`	enum	–	all = všechny fotony
`leaving-photon-length`	num	100	maximální délka stopy fotonu
`gamma`	num	1	gamma-korekce
`spectral`	bool	yes	zda se vykresluje spektrálně

`<scene>`

Je potomkem uzlu: `olav`
Je rodičem uzlu: `light-source, figure, sheet`

Definice objektů scény

Nemá atributy.

`<sheet>`

Je potomkem uzlu: `scene`
Je rodičem uzlu: `rectangle, file`

Stínítko (*direct*) nebo *snímek kamery* (*side*)

atribut	*typ*	*výchozí*	*hodnoty*
`type`	enum	–	side, direct
`disabled`	bool	no	zda je stínítko vypnuté

`<sphere>`

Je potomkem uzlu: `csg, figure`
Je rodičem uzlu: `-`

Geometrie koule – jsou dvě možnosti zápisu

atribut	*typ*	*výchozí*	*hodnoty*
`centre`	num3	–	střed koule
`radius`	num	–	poloměr

atribut	*typ*	*výchozí*	*hodnoty*
`pole`	num3	–	bod kružnice (pól)
`vector`	num3	(1, 0, 0)	vektor od pólu ke středu
`radius`	num	–	poloměr

`<threads>`

Je potomkem uzlu: `configuration`
Je rodičem uzlu: `-`

Nastavení vláken)

atribut	*typ*	*výchozí*	*hodnoty*
`light-threads`	num	1	počet vláken světelných zdrojů
`loop-threads`	num	1	počet výpočetních vláken

`<variable>`

Je potomkem uzlu: ***`jakéhokoliv`***
Je rodičem uzlu: `-`

Vyhodnocení výrazu. Je-li uveden atribut `name`, je výsledek výrazu uložen do proměnné a reportován na standardní výstup.

atribut	*typ*	*výchozí*	*hodnoty*
`name`	text	–	jméno cílové proměnné
`value`	text	–	libovolná hodnota nebo výraz

Příloha B

Obsah přiloženého CD

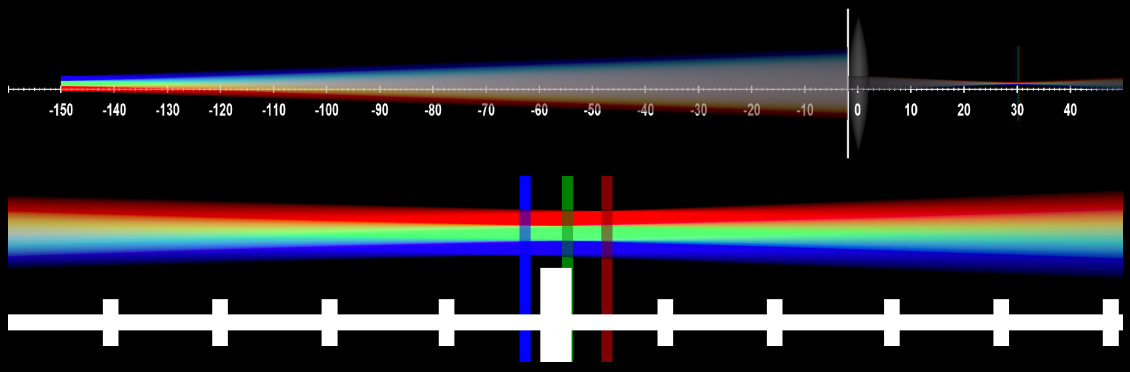
`doc/`	dokumentace
`doc/obj-model/`	dokumentace objektového modelu
`doc/BP.pdf`	text bakalářské práce
`lib/`	knihovny použité při testování na Windows
`scene/`	definiční soubory použitých scén
`scene/avi/`	výstup programu – animace
`scene/batch/`	dávky použité při spouštení výpočtu
`scene/bmp/`	výstup programu – obrázky
`scene/log/`	výstup programu – běhové informace
`scene/stats/`	statistiky pro vyhodnocení efektivity paralelizace
`src/`	zdrojový kód aplikace
`win32/`	soubory pro překlad na Windows
`win32/build/`	soubory projektu Visual Studio
`win32/lib/`	přeložené externí knihovny
`win32/lib-src/`	knihovny – projekty Visual Studio
`win32/include/`	hlavičkové soubory
`CMakeLists.txt`	definice projektu pro *cmake*

Příloha C

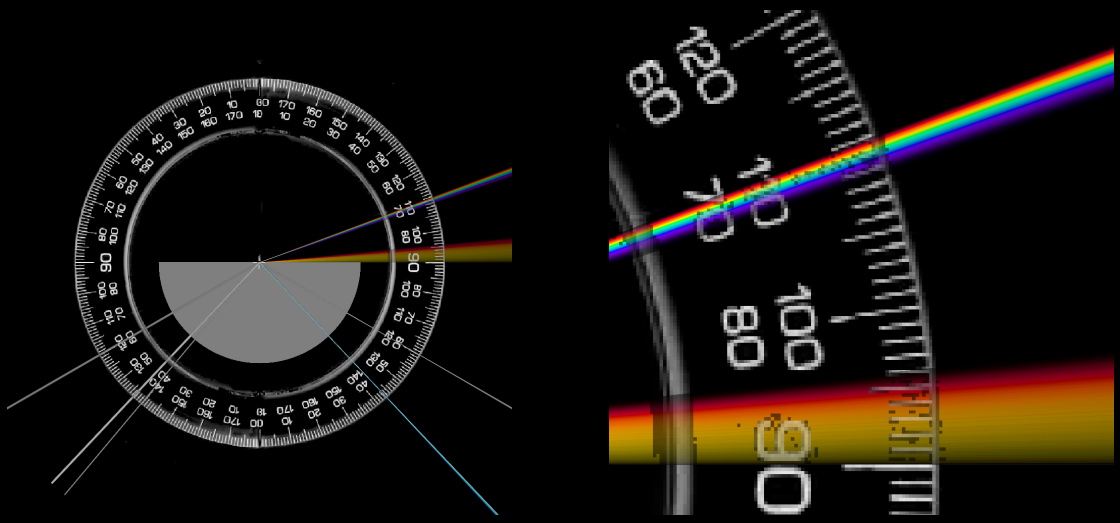
Obrazová příloha

Všechny uvedené příklady byly vytvořeny spektrálním simulátorem, obrázky nebyly dále graficky upravovány. Definiční soubory i vytvořené obrázky jsou k dispozici na přiloženém CD.

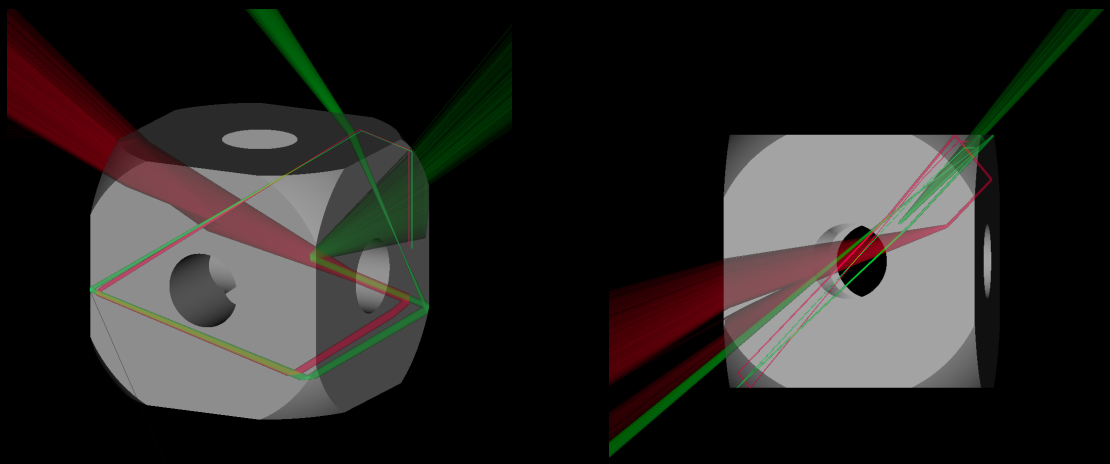
Obrázek C.1: **scene01.xml** Zobrazení spojnou čočkou, obrazové roviny R, G, B (viz odst. 7.1.1)



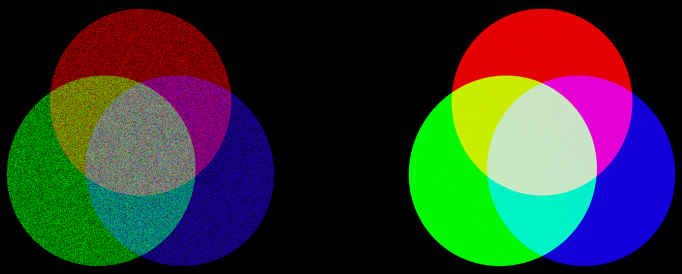
Obrázek C.2: **scene01.xml** Snímek kamery: a) celkový pohled na scénu, b) detail oblasti obrazových rovin



Obrázek C.3: **scene02.xml** Lom na rozhraní za vzniku spektra (viz odst. 7.1.2). Scéna je doplněna o dva další paprsky dopadající pod úhlem 40° a 60°, první je zcela rozložen do spektra, u druhého dochází k totálnímu odrazu. Na obrázku snímek kamery: a) celkový pohled na scénu, b) detail spektra

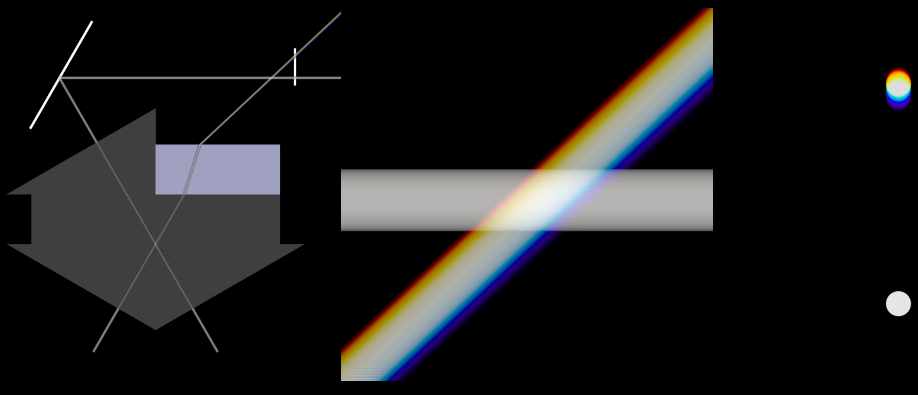


Obrázek C.4: **scene03.xml** CSG těleso, pohled na scénu z dvou různých úhlů (viz odst.7.1.3

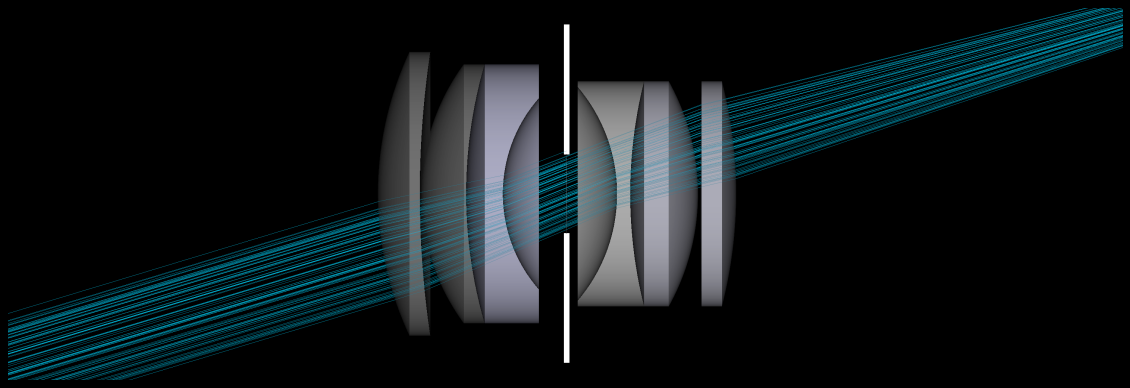


Obrázek C.5: **scene04.xml** Skládání barev (viz odst.7.1.4, výstup pro 1 mil. a 100 mil. fotonů při rozlišení 1000 × 1000 pixelů

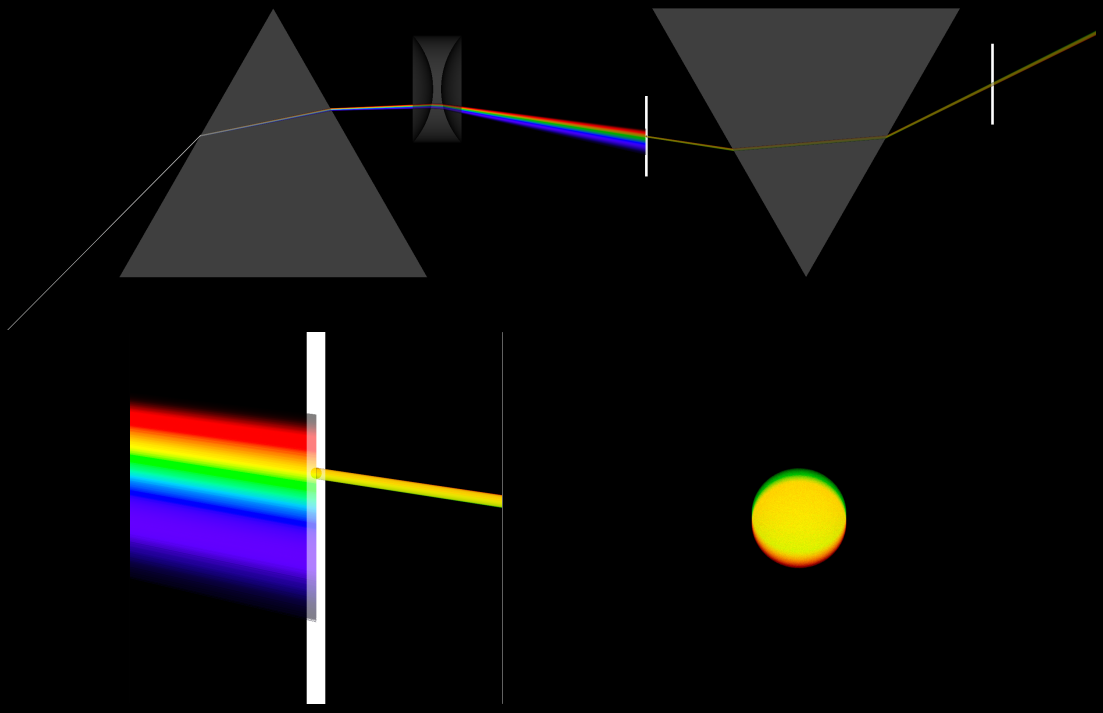
Obrázek C.6: **scene05.xml** Zobrazení bitmapy Taylorovým tripletem (viz odst.7.1.5); a) pohled na objektiv, b) originální bitmapa, c) výstup simulace



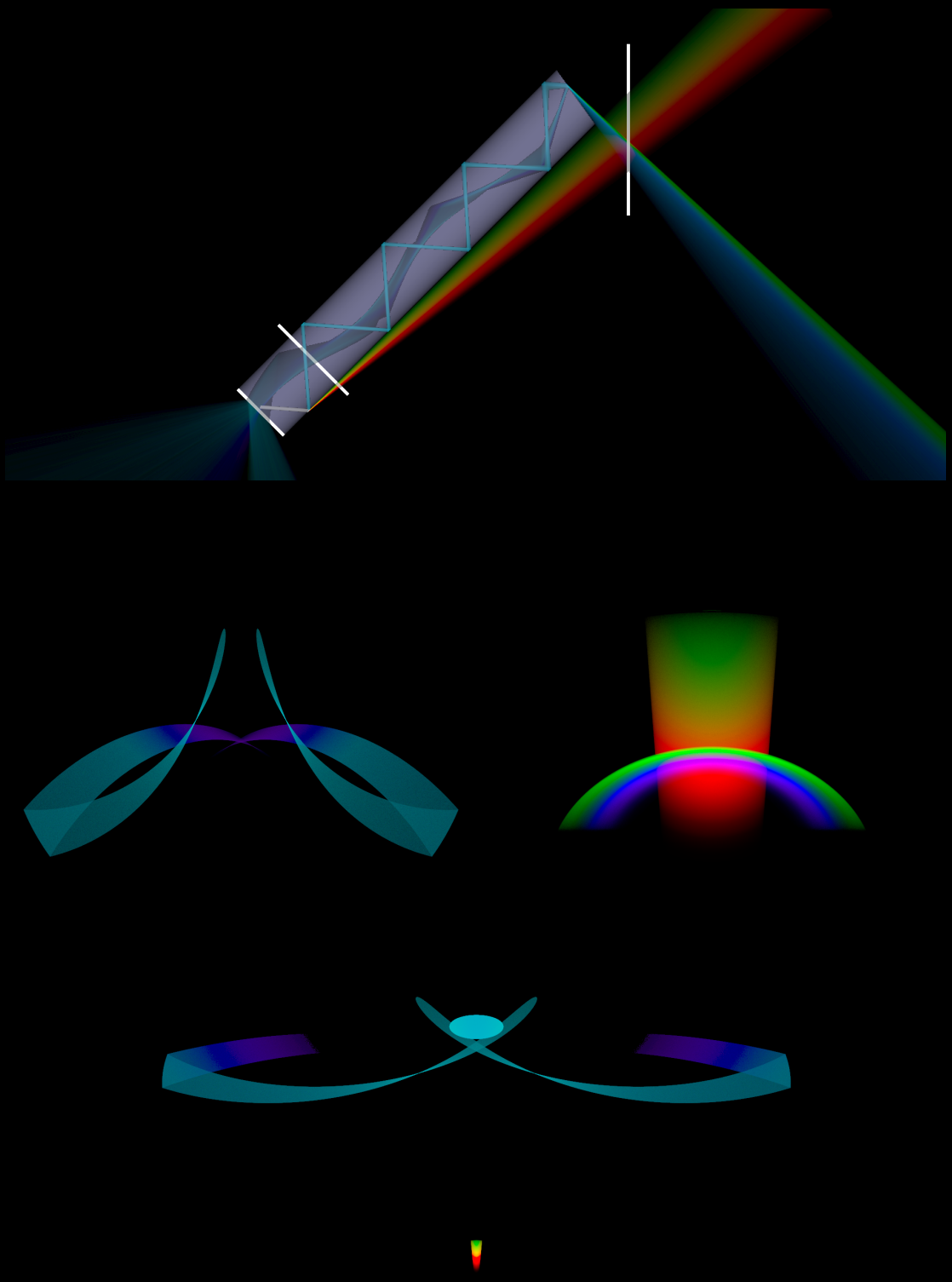
Obrázek C.7: **scene06.xml** Těleso slepené z několika menších těles (viz odst.7.1.6), a) pohled na scénu, b) detail paprsků, c) obraz na stínítku umístěném vpravo nahoře



Obrázek C.8: **scene08.xml** Dvojitý Gaussův objektiv (konstrukce přejata z [3])



Obrázek C.9: **scene07.xml** Jednoduchost spektrálních barev, klasický Newtonův pokus – světlo je rozloženo prvním hranolem, následně je ze spektra ponechán jen úzký svazek o jedné spektrální barvě, ta se již na dalším hranolu dále nerozkládá. Zabarvení okraje svazku je způsobeno příliš velkou světlostí clony a příliš širokým výchozím paprskem. Na obrázku a) celkový pohled na soustavu, b) detail clony, c) obraz na stínítku vpravo nahoře



Obrázek C.10: **scene09.xml** Zdroj světla se nachází ve středu spodní podstavy válce, světlo o složení D65 je směrováno tak šikovně, že polovina spektra (modřejší) při dopadu překročí mezní úhel a odráží se zpět do objemu válce, zbytek (červenější) se rozkládá mimo válec. Odražená část dále putuje válcem, opět se dělí na horní podstavě (pro dosažení vhodného úhlu je mírně zkosena), zde však nejen podle vlnové délky, ale i z důvodu různého úhlu dopadu. Rozptýlená část odchází zkosenou podstavou a skládá se s červeným spektrem z prvního dělení, odražená část se vrací válcem zpět a na spodní podstavě je konečně rozptýlena. Na obrázku a) celkový pohled na scénu, bílé čáry zobrazují polohu stínítek, b) obraz na stínítku ve spodní podstavě, c) obraz na stínítku vpravo nahoře, d) obraz na stínítku ve středu válce