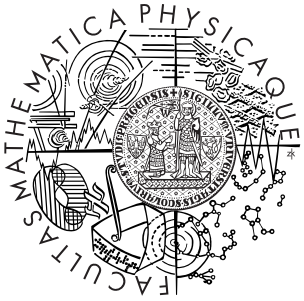
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Martin Bulant

Optická lavice

Kabinet software a výuky informatiky

Vedoucí bakalářské práce: RNDr. Josef Pelikán,
Studijní program: Informatika, programování

2010

Děkuji vedoucímu práce, RNDr. Josefu Pelikánovi, za poskytnuté konzultace, cenné rady a připomínky.

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

V Praze dne 5. srpna 2010 Martin Bulant

Obsah

Název práce: Optická lavice
Autor: Martin Bulant
Katedra (ústav): Kabinet software a výuky informatiky
Vedoucí bakalářské práce: RNDr. Josef Pelikán
e-mail vedoucího: pepca@cgg.mff.cuni.cz

Abstrakt: Předmětem této práce je vytvořit simulátor optické lavice. Simulátor bude zobrazovat rastrové obrázky pomocí soustavy čoček a výsledné obrazy bude zaznamenávat a ukládat opět do rastrových obrázků. Program je určen pro uživatele, kteří si chtějí vyzkoušet, jak fungují základní optické jevy. Pomocí programu bude možné odsimulovat jevy jako: lom a odraz světla, totální odraz světla a disperze světla. Další funkcí je zjišťování parametrů optické soustavy, ještě před jejím skutečným sestavením, například ohniskových vzdáleností.

Klíčová slova: optická lavice, sledování fotonů, simulace světla

Title: Optical bench
Author: Martin Bulant
Department: Department of Software and Computer Science Education
Supervisor: RNDr. Josef Pelikán
Supervisor's e-mail address: pepca@cgg.mff.cuni.cz

Abstract: The object of this work is to create optical bench simulator. The simulator will display bitmaps using system of lenses and the resulting images will record and then save in bitmaps again. Such program is for users, who want to try how basic optical phenomena work. With aid of the program,it will be possible to simulate phenomena such as: refraction and reflection of light, total reflection of light and light dispersion. Another function is colecting parameters of optical system before the system is actually assembled, for example focal lenght.

Keywords: optical bench, photon tracing, light simulation

Kapitola 1

Úvod

Výroba optických soustav je náročný proces, například pro objektivy. Stejné je to i s jejich návrhem. A právě při návrhu optických soustav má být nápomocen tento program. V současné době po navržení nové soustavy musí být její fyzikální vlastnosti velmi složitě analyticky spočítány, nebo musí být vyroben prototyp a vlastnosti musí být změřeny. To má za cíl změnit tento program, který po navržení soustavy je schopen odsimulovat chování světla v této soustavě pomocí promítání obrázků skrz tuto soustavu a zachytávání procházejícího světla na stínítka. Program by měl umět simulovat všechny základní optické jevy jako lom, odraz a disperze světla.

V první kapitole popisuji některé vlastnosti světla, které využívám při simulaci. Jsou to především, přímočaré šíření, odraz, lom a disperze. Dále pak uvádím různé způsoby uložení scény. Podrobně rozebírám CSG reprezentaci,tento způsob uložení jsem vybral pro reprezentování celé optické soustavy. Jako další pak uvádím možné druhy zobrazování scén a z nich především sledování fotonů (angl. photon tracing) a sledování světla (angl. light tracing). Nakonec této kapitoly uvádím způsob převodu barevných prostorů používaných v barevných obrázcích na spektrální reprezentaci využívanou tímto programem.

V druhé kapitole popisuji vytvořený program, jeho členění na hlavní celky a jejich funkci. Podrobněji rozebírám některé významnější třídy. Ukazuji způsob uložení optické soustavy v CSG stromu a metody, kterými se počítají průsečíky. Dále rozebírám způsob parsování XML souboru se schématem optické soustavy, jaké třídy k tomu používám a jakým algoritmem procházím vstupní soubor, respektive stromovou definici optické soustavy. Poslední částí je reprezentace obrázků.

Ve třetí kapitole ukazuji výsledky získané pomocí vytvořeného programu a popisuji získané obrázky. Nejdříve to je základní zobrazení bez čoček, dále pak zobrazení přes jednoduchou spojku, kde ukazuji vlastnosti získaného obrazu. Další ukázkou je totální odraz na hranolu. Poslední ukázkou je možný způsob zjišťování ohniskové vzdálenosti.

Kapitola 2

Principy zobrazování

V této kapitole popisuji některé vlastnosti světla, které využívám při simulaci. Jsou to především, přímočaré šíření, odraz, lom a disperze. Dále pak uvádím různé způsoby uložení scény. Podrobně rozebírám CSG reprezentaci,tento způsob uložení jsem vybral pro reprezentování celé optické soustavy. Jako další pak uvádím možné druhy zobrazování scén a z nich především sledování fotonů(angl. photon tracing) a sledování světla(angl. light tracing). Nakonec této kapitoly uvádím způsob převodu barevných prostorů používaných v barevných obrázcích na spektrální reprezentaci využívanou tímto programem.

2.1 Fyzikální vlastnosti světla

2.1.1 Popis světla

Viditelné světlo je součástí elektromagnetického spektra. To obsahuje různé druhy záření od rádiových vln, přes mikrovlny, viditelné světlo až po rentgenové a gama záření. Druh záření je určen jeho vlnovou délkou. Elektromagnetické záření vzniká oscilací elektricky nabitých částic. Rychlost jeho šíření je asi 300 000 km/s. Část elektromagnetického spektra, která je viditelná lidským okem je v rozsahu vlnových délek přibližně 380 až 720 nm. Konec odpovídající 720 nm odpovídá červené barvě opačný barvě fialové. V dalším textu se budu zabývat pouze viditelnou částí spektra.

Na světlo lze nahlížet několika různými způsoby a podle toho lze dělit i optiku. Ta se dělí na:

- Kvantovou

- Elektromagnetickou
- Vlnovou
- Paprskovou(Geometrickou)

jak uvádí Plášek a kol.[6]. Práce se bude zabývat především optikou geometrickou a částečně i vlnovou.

2.1.2 Interakce světla s prostředím

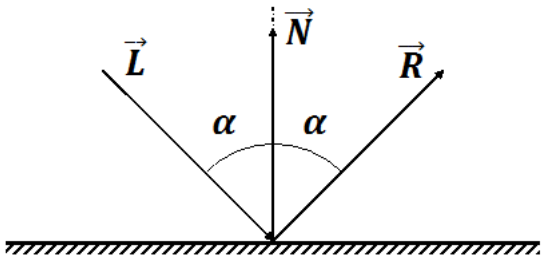
Po dopadu světla na povrch materiálu mohou nastat různé situace. V ideálním případě mohou nastat tyto 4 situace:

- `Zrcadlový odraz` – nastává při dopadu světla na lesklý povrch, jakým je například sklo. Platí zde zákon odrazu, kdy dopadající i odražený paprsek svírají s normálou tečné roviny v místě dopadu stejný úhel.
- `Difúzní odraz` – nastává při dopadu světla na matný povrch. Tuto situaci budu v této práci zanedbávat.
- `Lom dovnitř materiálu` – nastává u průhledných materiálů, příkladem takového materiálu je sklo, případně voda.
- `Difuzní rozptyl dovnitř materiálu` – nastává u průsvitných materiálů a tohoto případu budu využívat i pro pohlcení paprsku neprůsvitným materiálem.

Která z těchto situací nastane závisí především na vlastnostech materiálu, na který světlo dopadá. Jak jsem již uvedl, pro matné materiály nastvávají spíše difůzní jevy. Naprotitomu pro hladké a lesklé materiály se výsledné projevy velmi blíží ideálnímu, dokonale hladkému materiálu.

Index lomu

Nejvíce vlastností závisí na indexu lomu. Ten závisí na permitivitě a permeabilitě daného materiálu, jak uvádí Barčová a kol.[7]. Na index lomu lze pohlížet jako na konstantu platnou pro daný materiál. V takovém případě index lomu nezávisí na vlnové délce světla. Pro přesnější výsledky simulace lze ovšem index lomu brát jako závislý na vlnové délce světla, které dopadá na rozhraní dvou prostředí. Tuto závislost lze definovat pomocí Sellmeierova vzorce jak uvádí Glassner[2].



Obrázek 2.1: Odraz světla na rovinné ploše.

Zrcadlový odraz

Pro zrcadlový odraz platí jednoduchá pravidla z geometrické optiky pokud zanedbáváme Fresnelovy vztahy pro množství odraženého světla a toto množství nahradíme konstantou. Tato pravidla, jak uvádí Barčová a kol.[7] jsou odvozena z Huygensova principu a jsou to:

1. Odražený paprsek leží v rovině dopadu.
2. Při odrazu nedochází ke změně vlnové délky.
3. Úhel odrazu je roven úhlu dopadu.

Jak se odráží světlo od rovinné plochy je vidět na obrázku 2.1. Pro vektor odraženého paprsku platí rovnice:

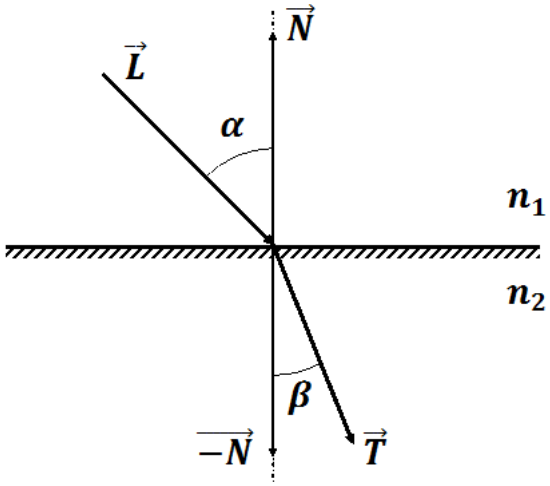
$$\mathbf{R} = 2\mathbf{N}(\mathbf{N} \cdot (-\mathbf{L})) + \mathbf{L} \tag{2.1}$$

jak uvádí Pelikán[5] pro opačně definovaný vektor L.

Dokonalý lom

I pro dokonalý lom lze použít jednoduchá pravidla vycházející z geometrické optiky. Tato pravidla, jak uvádí Barčová a kol.[7] jsou odvozena z Huygensova principu a jsou to:

1. Lomený paprsek leží v rovině lomu, která je totožná s rovinou dopadu.



Obrázek 2.2: Lom světla na rovinné ploše.

2. Při lomu dochází ke změně vlnové délky.

3. Úhel lomu není roven úhlu odrazu. Vztah úhlu dopadu a úhlu lomu je dán Shnellovým zákonem.

Pro lom světla na rozhraní dvou prostředí s různými indexy lomu platí Shnellův zákon jak uvádí Barčová a kol. [7] nebo Pelikán [5].

$$\frac{\sin\beta}{\sin\alpha} = \frac{n_1}{n_2} = n_{12} \tag{2.2}$$

Příklad takového lomu, je na obrázku 2.2, kde je zobrazen lom světla ke kolmici. Pro výsledný paprsek platí rovnice:

$$\mathbf{T} = [n_{12}(\mathbf{N} \cdot (-\mathbf{L})) - \sqrt{1 - n_{12}^2(1 - (\mathbf{N} \cdot (-\mathbf{L}))^2}] \cdot \mathbf{N} + n_{12} \cdot \mathbf{L} \tag{2.3}$$

jak uvádí Pelikán [5] pro opačně definovaný vektor L.

2.2 Zobrazovací metody prostorových dat

V počítačové grafice existuje celá řada metod, jak zobrazovat prostorová data. Tyto metody lze dělit dle mnoha kritérií, jedním z nich je podle věrnosti zobrazené scény. Máme tedy metody, které se snaží scénu zobrazit co

nejreálněji, těm se říká fotorealistické, a pak metody nefotorealistické. Fotorealistické metody se snaží scénu zobrazit tak, aby se co nejvíce podobala fotografii skutečné scény naproti tomu nefotorealistické metody scénu zobrazují s ohledem na specifické požadavky, například aby výsledek vypadal jako kresba, případně zobrazují jen tzv. drátové modely. V této práci se zaměřím především na metody fotorealistické, protože budu provádět simulaci, která se snaží, aby její výsledky byly co možná nejvíce podobné realitě.

Globální zobrazovací metody jsou takové, ve kterých se při zobrazování jednoho objektu ve scéně projevuje i přítomnost dalších objektů. Tyto další objekty způsobují především změnu osvětlení zobrazovaného objektu, mohou množství dopadající světla na objekt snížit tím, že ho například zastíní, nebo naopak zvýšit, pokud se světlo ze světelného zdroje odrazí například od zrcadla a dopadne na zobrazovaný objekt. Tím se liší od lokálních metod, které řeší nejvýše viditelnost, jak uvádí Žára a kol.[3]. Rozeberu 2 rozdílné zástupce globálních zobrazovacích metod.

2.2.1 Metody konečných prvků

První metodou je metoda takzvaných konečných prvků. To je metoda, ve které se scéna rozdělí na určitý počet ploch, nejčastěji trojúhelníků a mezi nimi se vyřeší zobrazovací rovnice. Po vyřešení zobrazovací rovnice už má každá plocha spočítané globální osvětlení a stačí tuto plochu jednoduše zobrazit.

2.2.2 Sledování paprsku

Druhou metodou, jejíž poznatky budu využívat v této práci je sledování paprsku. Tato metoda obecně vychází z principu geometrické optiky, že se světlo šíří v paprscích. Tyto paprsky se ve vakuu šíří přímým směrem. A jejich směr se mění pouze při dopadu na povrch tělesa, případně při průchodu opticky aktivním materiálem. Všechny metody, které používají sledování paprsku se vyznačují tím, že se vezme paprsek, který je definován bodem a směrem a spočítá se jeho průsečík se scénou. Pak už se jednotlivé metody liší v tom, jaké je další chování poté, co se tento průsečík získá.

Metody směrem od pozorovatele

Metody směrem od pozorovatele vycházejí z principu dírkové kamery. Ovšem aby nevznikal převrácený obraz, je průmětna předsunuta otvoru. A původní

funkce otvoru se nahradí tím, že všechny paprsky budou vysílány z jednoho bodu, který je v místě již nevyužívaného otvoru v dírkové kameře. Více informací poskytuje Shirley[4]. Obraz se získá tak, že se jednotlivými obrazovými body průmětny posílají paprsky do scény. Výpočet zde probíhá rekurzivně. Paprsky jsou vyslány do scény, kde interagují s tělesy ve scéně obsaženými. Po nárazu do tělesa paprsek zjistí, jestli je dané místo osvětleno od nějakého zdroje světla přímo, v tom případě se spočítá hodnota osvětlení, pokud už nebyla spočítána dříve, například pomocí metody konečných prvků. Paprsek se v průsečíku pak dále od tělesa odrazí, případně se zalomí dovnitř tělesa a postup se opakuje. Při zpětném průchodu se vrací informace o barvě a jasu, takže po návratu do bodu průmětny zanese hodnota barvy a radiance získaná rekurzivním sledováním jednoho paprsku.

Mezi metody vycházející směrem od pozorovatele se řadí rekurzivní sledování paprsku a sledování cesty. Přičemž tyto dvě se od sebe liší především schopností zobrazovat difúzní odrazy. Sledování paprsku při nárazu na těleso vysílá většinou dva další paprsky. Těmi jsou zrcadlový odraz a paprsek lomený dle Shnellova zákona. Sledování cesty po nárazu na těleso vysílá paprsků více do náhodných směrů, čímž lépe zobrazuje matné odrazy a matné lomy. Více podrobností píše Žára a kol. [3].

Metody směrem od zdroje světla

Metody směrem od zdroje světla jsou duální k metodám od pozorovatele. Od metod směrem od pozorovatele se liší především tím, že paprsky zde nesou výkon. S paprsky vyzářenými ze světelného zdroje se zachází jako by to byly skutečné světelné paprsky dle pravidel pro geometrickou(paprskovou) optiku. Mezi tyto metody patří sledování fotonů(photon tracing) a sledování světla(light tracing). Přičemž sledování fotonů se příliš nepoužívá, jelikož umí zobrazovat pouze zrcadlové odrazy a lom dovnitř materiálu. Sledování světla je duální metoda ke sledování cesty. Po každém nárazu paprsku do scény se náhodně zvolí další směry, kterými se paprsek bude šířit a podle zobrazovací rovnice se mu dá příslušné množství energie.

Dvojsměrné metody zobrazování

Jak píše Žára [3], dvojsměrné metody v sobě spojují výhody obou předchozích způsobů zobrazování. Cesta se počítá jak od pozorovatele, tak od zdroje, přičemž počítání cesty od zdroje je výhodné pro scény se složitým osvětlením a pro výpočet kaustik. Naproti tomu cesta od pozorovatele umožňuje

snadnější určování přesnosti a snižuje šum výsledného obrázku.

2.3 Způsob uložení scény

Způsoby uložení scény v počítačové grafice lze rozdělit dle toho, jestli je těleso prezentováno svým objemem, nebo pouze povrchem. Přičemž tělesem rozumíme spojitou strukturu, kterou reprezentujeme množinou bodů, tyto body ještě může rozdělit na vnitřní a porchové, přičemž vnitřní body sousedí s jinými vnitřními body, nebo body povrchovými. A body povrchové sousedí alespoň s jedním vnitřním bodem jak uvádí Žára a kol. [3]. Povrchovým bodům se říká body hraniční.

2.3.1 Povrchové reprezentace

Jak uvádí Žára a kol. [3] jejich použití je vhodnější pro pozdější zpracování scény. Povrchové reprezentace lze ještě rozdělit na reprezentace pomocí vrcholů, hran a ploch a na tzv. CSG(Constructive Solid Geometry) reprezentace, které jsou využívány především v CAD systémech. Jelikož ve svém programu využívám CSG reprezentaci, budu dále popisovat pouze tu.

CSG reprezentace je tvořena stromovou strukturou, jejíž vnitřní uzly přestavují operátory, a listy představují elementární tělesa. Operátory jsou sjednocení(union), průnik(intersection) a rozdíl(difference). Operátory jsou definovány jako binární, ale například sjednocení a průnik lze použít i jako n-ární operátor. Z toho vyplývá, že strom, který scénu reprezentuje nemusí být nutně binární. CSG strom scény se používá ve dvou základních variantách. První z nich, je varianta, kdy jsou jednotlivé transformační matice uloženy i ve vnitřních uzlech. Tato reprezentace je vhodná pro scény, ve kterých dochází k pohybu těles. Protože při pohybu podstromu stačí měnit matice pouze v uzlu, který je kořenem tohoto podstromu. Druhou variantou je případ, kdy jsou transformační matice uloženy pouze v listech. Tato varianta je vhodná především pro statické scény. Její výhodou je, nižžší počet potřebných maticových transformací paprsků při zjišťování jaký je průsečík paprsku se scénou. Z toho důvodu je tato varianta využívána při simulaci.

2.3.2 Objemové reprezentace

Pro úplnost uvádím i objemový způsob reprezentace těles. Tento způsob používá reprezentaci tělesa pomocí rastru, který je v tomto případě dělen

na voxely. Přičemž těleso je pak reprezentováno skupinou voxelů z dané trojrozměrné mřížky. Při zobrazování těchto dat, se objemová reprezentace převádí na reprezantaci povrchovou, se kterou se lépe pracuje.

2.4 Převod RBG na spektrum

Jak uvádí Glassner[1], pro převod barvy uložené ve formátu RGB do spektrální podoby lze použít odhadnutí průběhu intenzit pomocí funkcí 2.4, které pracují se spektrem v rozsahu od 380 do 720 nm.

$$\begin{aligned} F_1(\lambda) &= 1.0 \\ F_2(\lambda) &= \frac{1}{2}[1 + \sin(2\pi\frac{\lambda - 380}{340})] \\ F_3(\lambda) &= \frac{1}{2}[1 + \cos(2\pi\frac{\lambda - 380}{340})] \end{aligned}$$

K těmto funkcím se pak hledají koeficienty $\mathbf{W} = [w_1, w_2, w_3]$, díky kterým pak získám spektrální reprezentaci hledaná barvy při použití vzorce 2.4.

$$C(\lambda) = w_1 F_1(\lambda) + w_2 F_2(\lambda) + w_2 F_2(\lambda). \tag{2.4}$$

Hodnoty vekturu $\mathbf{W}$ získám z rovnice 2.5, kterou uvádí Glassner[1]. $\mathbf{R}$ v této rovnici je barva uložená ve formátu RGB, $\mathbf{M}$ je převodní matice mezi barevnými prostory RGB a XYZ a matice D je definována v rovnici 2.6. Funkce $\bar{x}$, $\bar{y}$ a $\bar{z}$ jsou funkce takzvaného standardního pozorovatele.

$$\mathbf{W} = \mathbf{R}(\mathbf{DM})^{-1} \tag{2.5}$$

$$\mathbf{D} = \begin{pmatrix} \int_{\lambda \in R_v} F_1(\lambda)\bar{x}(\lambda)d\lambda & \int_{\lambda \in R_v} F_1(\lambda)\bar{y}(\lambda)d\lambda & \int_{\lambda \in R_v} F_1(\lambda)\bar{z}(\lambda)d\lambda \\ \int_{\lambda \in R_v} F_2(\lambda)\bar{x}(\lambda)d\lambda & \int_{\lambda \in R_v} F_2(\lambda)\bar{y}(\lambda)d\lambda & \int_{\lambda \in R_v} F_2(\lambda)\bar{z}(\lambda)d\lambda \\ \int_{\lambda \in R_v} F_3(\lambda)\bar{x}(\lambda)d\lambda & \int_{\lambda \in R_v} F_3(\lambda)\bar{y}(\lambda)d\lambda & \int_{\lambda \in R_v} F_3(\lambda)\bar{z}(\lambda)d\lambda \end{pmatrix} \tag{2.6}$$

Kapitola 3

OpticalBench

V této kapitule popisuji vytvořený program. Ukazuji jeho hlavní součásti a podrobněji se o nic rozepisuji. Zvláštní pozornost věnují vnitřní reprezentaci optické soustavy a metodám práce s ní, dále pak načítání optické soustavy ze schématu ve formátu XML. A v poslední částí rozebírám vnitřní reprezentaci obrázků, jejich možné typy, a metody, kterými se s těmito obrázky pracuje.

3.1 Popis funkce

Po spuštění program zpracuje parametry. Pokud se mezi parametry nachází parametr -d, program se přepne do dávkového režimu a ostatní parametry ignoruje. V opačném případě z parametru -s soubor načte název souboru, se schématem optické soustavy a z parametru -p pocet, počet paprsků, které mají vycházet z jednoho pixelu. Po zpracování parametrů třída Model načte schéma optické soustavy z daného XML dokumentu. Před započetím načítání se ještě zkontroluje, zda schéma soustavy odpovídá DTD schématu. Model se načítá pomocí SAX parseru. To znamená že parser prochází dokument a při nalezení elementu zavolá mnou definovanou funkci. Během načítání schématu soustavy se rovněž načítají vstupní obrázky. Po úspěšném načtení celého XML se začne provádět simulace.

Simulace se provádí po jednotlivých vstupních obrázcích, kdy se každý obrázek projde pixel po pixelu a pro každý pixel se provede simulace vyzářeného světla. Po vyzáření světla vstupním obrázkem se testují průniky paprsků se zbytkem optické soustavy. Pokud se najde průsečík paprsku s optickou soustavou. Spočítá se normála na rozhraní dvou materiálů v místě dopadu a zjistí se oba materiály. Poté se zpracuje zjištěný průsečík. Výsled-

kem zpracování průsečíku může být další paprsek, který vznikl lomem, nebo odrazem v místě průsečíku. S tímto se pak pokračuje jako by byl vyzářen z obrázku dokud bude mít průsečíky s optickou soustavou nebo nebude pohlcen. Během této simulace se světlo, které projde stínítky zaznamená. Po ukončení simulace pro všechny obrázky se zachycené světlo na stínítkách převede do obrázků a ty se uloží.

3.2 Reprezentace optické soustavy

Jak jsem již uvedl, pro reprezentaci celé scény používám CSG reprezentaci. Ta je velice vhodná, jelikož skutečné čočky pro sférickou optiku lze reprezentovat průnikem, případně rozdílem koule s poloprostorem, nebo další koulí. Jednotlivé uzly CSG stromu jsou vzájemně propojeny pointry ve směru od rodiče k potomkovi. Soustava je rozdělena na několik základních částí, jsou jimi. Čočky, zdroje, stínítka a clonky. Čočky a clonky jsou reprezentovány binárními CSG stromy. Tyto stromy jsou drženy v kořeni spolu se zdroji i stínítky.

3.2.1 Použité třídy

Pro uložení optické soustavy jsou využívány tyto třídy:

UzelStromu

Abstraktní třída, která reprezentuje uzel, případně list CSG stromu. Od ní jsou odvozeny dva typy vnitřních uzlů, jeden typ listu a kořen celého stromu. Deklaruje tyto virtuální metody:

- `NavratovaHodnota& ProtniRychle(Paprsek p)`
- `t_genPaprsky Generuj(GeneratorPaprsku & gen, int index)`
- `void Uprav(Matice transformace,Matice inverzni)`

První metoda se volá při testování průniku paprsek x CSG strom. Druhá při inicializaci výpočtu, kdy se stará o vygenerování paprsků ze vstupních obrázků. Poslední zajišťuje převod stromu do statické podoby, to znamená, že transformační matice zůstanou pouze v listech stromu.

KorenStromu

Třída odvozená od třídy `UzelStromu`, která drží celou scénu optické soustavy pohromadě pomocí sjednocení. To je zajištěno tím, že si drží seznam odkazů typu `UzelStromu`.
Při zavolání metody `NavratovaHodnota& ProtniRychle(Paprsek p)` se metoda zavolá na všechny podstromy, čímž se získá seznam setříděných seznamů průsečíků paprsku s podstromy. Tenty seznamy se slijí metodou:
`NavratovaHodnota SlijSeznamy(vector<NavratovaHodnota* >& s)`,
která využívá druhou část algoritmu mergesort. Přičemž slévání probíhá naráz pro všechny seznamy a aktuální prvky jsou drženy v haldě.

VnitrniUzelA

První část vnitřního uzlu CSG stromu. Udržuje v sobě množinovou operaci a následníky, kterými jsou odkazy na `VnitrniUzelB`.
Při zavolání metody `NavratovaHodnota& ProtniRychle(Paprsek p)` zajišťuje její zavolání na potomky a potom zajišťuje slití seznamů průsečíků získaných od potomků do výsledného seznamu platného pro tento uzel, to obstarává metoda
`void SlijSeznamy(NavratovaHodnota& nh1, NavratovaHodnota& nh2)`.

VnitrniUzelB

Druhá část vnitřního uzlu CSG stromu. Tato třída plní pouze funkci prostředníka, jelikož obsahuje pouze odkaz na další uzel CSG stromu.

ListStromu

Obsahuje v sobě odkaz na jedno ze základních těles a dále transformační matici a k ní inverzní transformační matici. Při zavolání metody
`void Uprav(Matice transformace,Matice inverzni)` se do těchto matic uloží matice získané při průchodu celým stromem do hloubky. Tyto matice pak slouží jako transfomační matice pro transformoce mezi lokálními souřadnicemi drženého tělesa a světovými souřadnicemi.

Operace

Abstraktní třída pro binární operaci. Jsou od ní odvozeny třídy `Sjednoceni`, `Prunik` a `Rozdil`, které zabezpečují jejich názvy dané binární bitové operace.

Pomocí těchto operací se určuje výsledek slévání dvou seznamů průsečíků, které určují intervaly na paprsku.

Teleso

Abstraktní třída reprezentující základní tělesa. Možná základní tělesa jsou:

- `Koule` pro jednotkovou kouli se středem v počátku soustavy souřadnic.
- `Krychle` pro jednotkovou krychli se středem v počátku soustavy souřadnic.
- `Poloprostor` pro poloprostor, který má hraniční rovinu `x=0` a objem má směrem k záporné části osy x.
- `Valec` pro nekonečný válec o poloměru 1 jednotka, jehož osou je osa x soustavy souřadnic.
- `Kuzel` pro dvojitý nekonečný kužel s vrcholem v počátku soustavy souřadnic, jehož osou je osa x soustavy souřadnic.
- `RotacniParaboloid` pro rotační paraboloid s vrcholem v počátku soustavy souřadnic a s objemem směřujícím v kladném směru osy x.
- `RotacniHyperboloid` pro rotační hyperboloid, jehož osou je osa x oustavy souřadnic.

Definuje virtuální metody:

- `void Protni(NavratovaHodnota& pruseciky,Paprsek p)`
- `t_genPaprsky Generuj(GeneratorPaprsku & gen, int index)`

Základní tělesa jsou jediné třídy, které skutečně počítají průsečík. Ostatní třídy v CSG stromu pouze zprostředkovávají volání metod a případně slévají seznamy zjištěných průsečíků.

3.3 Načítání XML

Parsování XML dokumentu probíhá pomocí SAX parseru xerces. Tedy systémem událostí a reakcí na ně. Pro zpracovávání událostí vzniklých při parsování dokumentu slouží třída `MySAX2Handler` a třídy `MXML*`, celé parsování zajišťuje třída `XMLParser`. Třída `MySAX2Handler` si při inicializaci vytvoří seznam všech elementů použitých pro definici optické soustavy. Dále si vytvoří zásobník pointrů na třídu `MXMLObjekt`.

Při nalezení začátku elementu je zavolána metoda `startElement( ... )`, jejíž parametr `localname` udává název aktuálně začínajícího elementu. Pro toto jméno se v seznamu nalezne odpovídající třída a pointr na její instanci se vloží na vrchol zásobníku.

Pokud element obsahuje jen řetězec znaků, volá se `characters( ... )`, která zavolá metodu `ZpracujText(string hodnota)` u vrcholu zásobníku a jako parametr dostane obsah elementu, tedy řetězec znaků.

Při nalezení konce elementu je zavolána metoda `endElement( ... )`, která ze zásobníku vezme vrchol a pokud zásobník není prázdný, na novém vrcholu zásobníku zavolá metodu `ZpracujHodnotu(MXMLObjekt* hodnota)` a jako parametr jí předá původní vrchol zásobníku.

3.3.1 Knihovna xerces

Pro parsování XML je použíta knihovna xerces-C++ 3.1.1. Informace o této knihovně lze nalézt na: `http://xerces.apache.org/xerces-c/index.html`. Tato knihovna je šířena pod licencí: Apache Licence verze 2.0 dostupné zde: `http://www.apache.org/licenses/LICENSE-2.0`

3.3.2 Použité třídy

XMLParser

Třída zajišťující parsování zadaného XML dokumentu. Obsahuje v sobě reference na seznamy materiálů, vstupních a výstupních obrázků a na prostředí. Všechny tyto položky jsou uloženy ve třídě model, takže při parsování mohou být přímo načítány. Parsování dokumentu se spouští zavoláním metody `int Parse(string fileName)`. Po skončení parsování XML dokumentu se pointr ukazující na CSG model optické soustavy získá pomocí metody `UzelStromu* ZiskejModel()`.

MySAX2Handler

Třída pomáhající při parsování dokumentu. Její metody jsou volány pro zpracování jednotlivých událostí při parsování XML. Těmito metodami jsou:

- `void startElement(const XMLCh* const uri, const XMLCh* const localname, const XMLCh* const qname, const Attributes& attrs)`

  – zpracovává začátek elementu, název právě začínajícího elementu je uložen v parametru localname

- `void endElement(const XMLCh* const uri, const XMLCh* const localname, const XMLCh* const qname)}`

  – zpracovává konec elementu, název právě ukončeného elementu je uložen v parametru localname

- `void characters(const XMLCh* const ch,const XMLSize_t l)` – zpracovává obsah elementu, to znamená znaky uvnitř elementu, tato metoda se volá, pokud už daný element neobsahuje vnořené elementy.

- `void error(const SAXParseException&)` – zpracovává výjimky a nezávažné chyby, které se vyskytly při parsování XML dokumentu

- `void fatalError(const SAXParseException&)` – zpracovává chyby, které se vyskytly při parsování XML dokumentu.

MXML*

Do této skupiny spadají všechny třídy začínající písmeny „MXML“. Jedná se o dočasné třídy, sloužicí k uchovávání dočasných informací během parsování XML dokumentu. Všechny tyto třídy mají společného předka, kterým je třída `MXMLObjekt`, tato třída definuje dvě virtuální metody:

- `ZpracujText(string hodnota)` – slouží pro zpracování obsahu elementu, kterým je řetězec znaků

- `ZpracujHodnotu(MXMLObjekt* hodnota)` – slouží pro zpracování vnořeného elementu

3.4 Reprezentace obrázků

Obrázky reprezentuji jako dvojrozměrná pole obrazových bodů, které mohou být různého typu. Pro reprezentaci obrázků jsou použity následující třídy:

Obrazek

Tato třída se stará o reprezentaci rastrových obrázků, které jsou používány v programu. Tyto obrázky se používají jak pro zdrojové, tak pro cílové obrázky, neboli stínítka. Skládá se z obrazových bodů, které jsou typu `ObrazovyBod`.

Obrazovy bod

Abstraktní třída, od které dědí konkrétní třídy reprezentující obrazový bod. Bod je reprezentován těmito třídami: `GrayScale`, `RGB` a `Spektrum`. Uchovává v sobě informaci o intenzitě a jasu. Deklaruje virtuální metody:

- `PrevedInaJ()` – stará se o převod intenzity na jas před ukládáním obrázku.
- `Uloz(ofstream & kam)` – slouží k uložení jednoho obrazového bodu.
- `Generuj(t_sezDvojVektor& p, Bod s, Bod v1, Bod v2, int k)` – vytváří paprsek vycházející z daného obrazového bodu. Zajišťuje mu správnou intenzitu, barvu a vlnovou délku.
- `ZanesHodnotu(Paprsek smer_,Paprsek normala_)` – slouží k uložení hodnoty intenzity paprsku, který protnul tento pixel.

Kapitola 4

Simulace zobrazení pomocí soustavy čoček

V této kapitole ukazuji výsledky získané pomocí vytvořeného programu a popisuji získané obrázky. Nejdříve rozebírám kvalitu získaných obrázků a rychlost jakou simulace probíhá. Pak ukazují nejjednoduší soustavu bez čoček, dále pak zobrazení pomocí spojky, kde ukazuji vlastnosti takového obrazu. Další ukázkou je totální odraz na hranolu. Poslední ukázkou je možný způsob zjišťování ohniskové vzdálenosti u složitější soustavy čoček.

Tabulka 4.1: Doby simulace pro scénu bez odrazů

Počet paprsků	Doba simulace (s)
1	8
5	14
10	22
50	78
100	155
500	770
1000	1572

Tabulka 4.2: Doby simulace pro scénu s odrazy

Počet paprsků	Doba simulace (s)
1	8
5	19
10	37
50	212
100	456

4.1 Kvalita výstupu a rychlost simulace

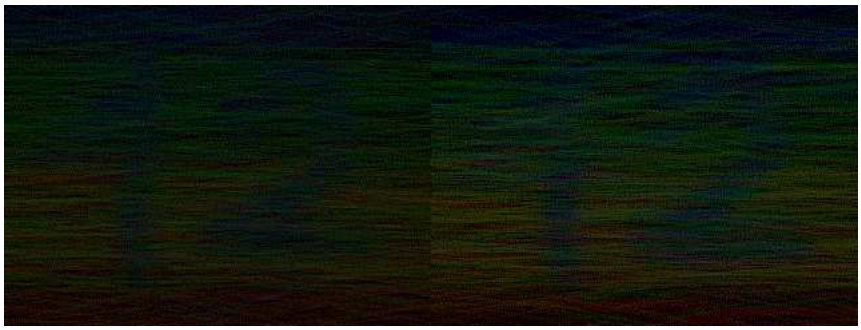
Kvalitu výstupu a rychlost provádění simulace budu testovat na optické soustavě využité při ukazování vlastností zobrazení přes spojku. Budu používat vstupní obrázek s rozlišením 320x240 pixelů. Výstupní obrázky jsou stejného rozlišení a jejich počet je 27. Simulaci provedu nejdříve bez odrazů na porvších materiálů a poté s odrazy nastavenými na 20%.

Po provedení simulací s různými počty paprsků jsem naměřil výsledky uvedené v tabulce 4.1 pro simulaci bez odražených paprsků. Pro malé počty paprsků probíhá simulace velice rychle, ovšem výsledné obrázky tomu odpovídají jak je vidět na obrázku 4.2.

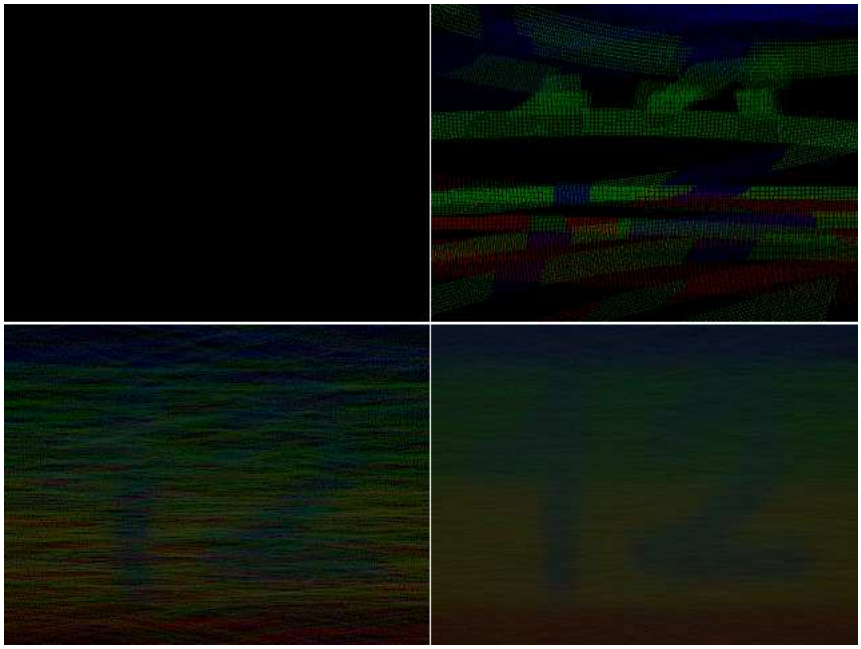
Naproti tomu, při porovnání výsledných obrázků získaných s odrazy a bez odrazů mezi nimi přiliš velký rozdíl není, na obrázcích získaných s odrazy, je sice více šumu, ale díky menšímu množství světla dopadajícího na získaný obrázek se zdá, že je zašuměný méně viz. obrázek 4.1. Ovšem doby výpočtů simulací s odraženými paprsky jsou mnohem delší než při výpočtu bez odrazů. Doby výpočtů s odrazy jsou vidět v tabulce 4.2.

4.2 Zobrazování bez čoček

Prvním příkladem je velmi jednochá scéna složená pouze z jednoho zdroje a jednoho stínítka. Obě roviny jsou rovnoběžné a umístěné 10 jednotek od sebe. Schéma této optické soustavy je uloženo v souboru `priklad1.xml`. Jako zdrojový obrázek slouží 4.3. Tento obrázek v sobě obsahuje různé barvy a vrchní část je mnohem světlejší než část spodní. Výsledek této simulace je vidět na obrázku 4.4. Na tomto obrázku jsou trošku znatelné vodorovné čáry které byly v původním obrázku, ale ztratila se veškerá informace o barvě. Je vidět, že dolní část obrázku je trochu tmavší než část vrchní, to je způsobeno



Obrázek 4.1: Srovnání obrázků získaných s odrazy (vlevo) a bez odrazů (vpravo).



Obrázek 4.2: Srovnání obrázků získaných simulací pomocí 1 (vlevo nahoře), 10 (vpravo nahoře), 100 (vlevo dole) a 1000 paprsků (vpravo dole).

Obrázek 4.3: Vstupní obrázek pro soustavu bez čoček.

tím, že zdroj od stínítka není příliš vzdálen.

4.3 Jednoduchá spojka

Druhým příkladem je už mnohem zajímavější zobrazování pomocí jedné čočky. Vybral jsem sférickou čočku tvořenou průnikem dvou koulí o stejném poloměru. Schéma této optické soustavy je uloženo v souboru `priklad2.xml`. V tomto schématu je použit jeden zdroj a více stínítek. Tím docílím toho, že budu moci ukázat, jak by vypadal výsledný obrázek v různých vzdálenostech od zdroje a čočky. Při výpočtu ukázkových obrázků jsem použil 1000 paprsků na pixel.

Vstupním obrázkem tohoto příkladu je obrázek 4.5. Na tomto obrázku jsou 4 jasné vodorovné pruhy, přičemž červený pruh je nahoře a modrý dole. Přes prostřední pruhy je text napsaný vzhůru nohama. Na obrázku 4.6 je vidět bílý šum nahoře přecházející do odstínů červené, dole do odstínů modré, tedy stejně jako ve vstupním obrázku, tento obrázek je získaný ještě před čočkou. Obrázek 4.7 je získaný kousek za čočkou, na tomto obrázku je dobře vidět kruhovitost čočky. Dále zde šum, který je uprostřed přechází

Obrázek 4.4: Obrázek získaný při simulaci bez čoček.

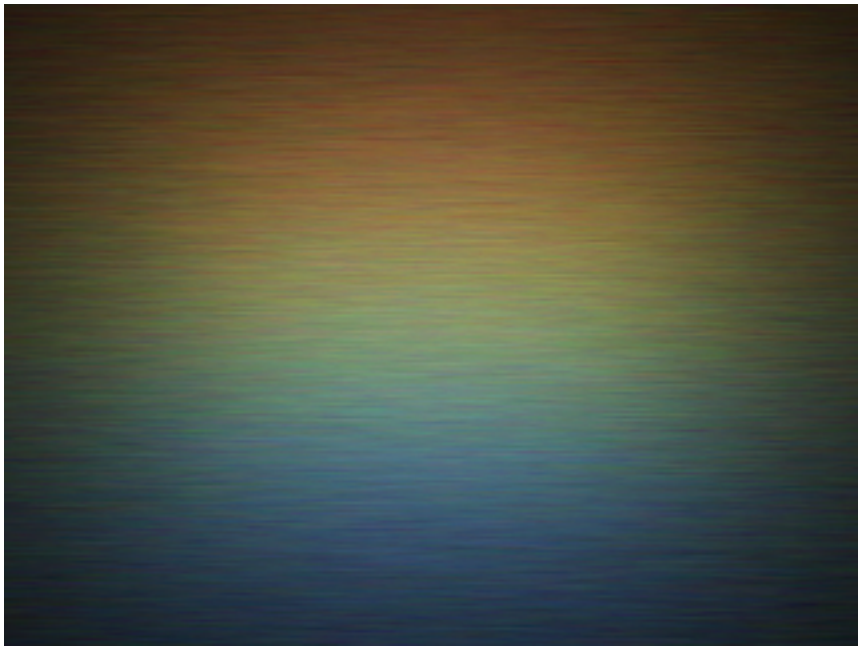
do modré barvy nahoře a do červené dole, což je opačně oproti vstupnímu obrázku.

Na dalších dvou obrázcích už je jasně patrná původní pruhovitá struktura vstupního obrázku. Opět jsou barvy v opačném pořadí než ve vstupním obrázku. Na prvním z nich, tedy na obrázku 4.8 jsou dobře patrné pruhy, naproti tomu text uprostřed je nečitelný, z číslic zbyly pouze velmi rozmazané obrysy. Naproti tomu na obrázku 4.9 už jsou číslice čitelné dobře, ale bohužel se ztrácí jasnost barevných pruhů, to je způsobeno větším zvětšením a tím, že je tento obrázek poněkud tmavší než předchozí. Na tomto obrázku je text ve správné poloze, tedy převrácený vůči vstupnímu obrázku. Jak už jsem uvedl, obrázek je též zvětšený.

4.4 Totální odraz

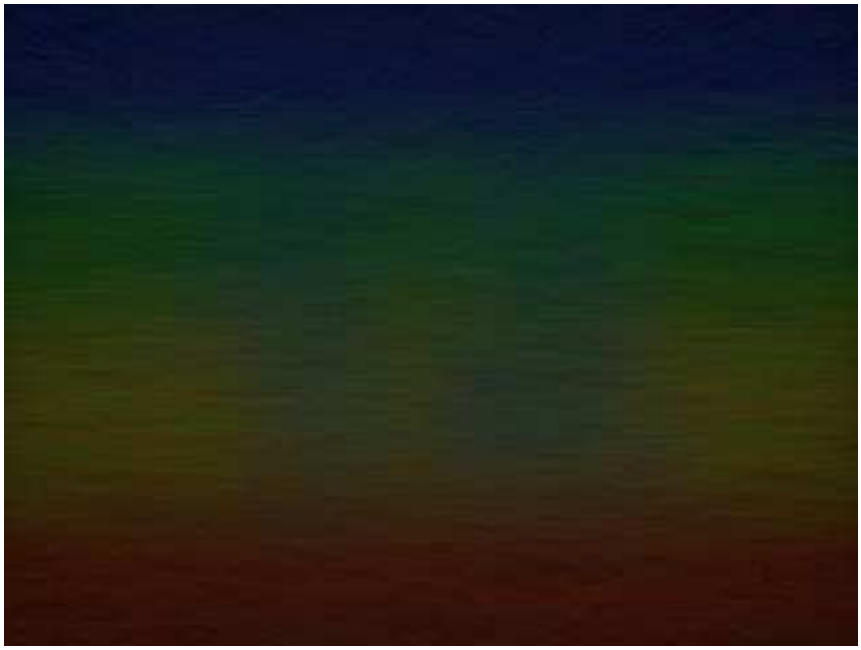
V tomto příkladě ukáži, jak se světlo zalomí na jedné stěně hranolu a na druhé se už pouze odrazí, a to i přesto, že odrazivost je nastavena na nulu pro všechny použité materiály. V tomto příkladě využiji schéma ze souboru `priklad3.xml`. Opět použiji jako zdroj obrázek 4.5. Zobrazovat budu přes

Obrázek 4.5: Obrázek zobrazovaný přes spojku.

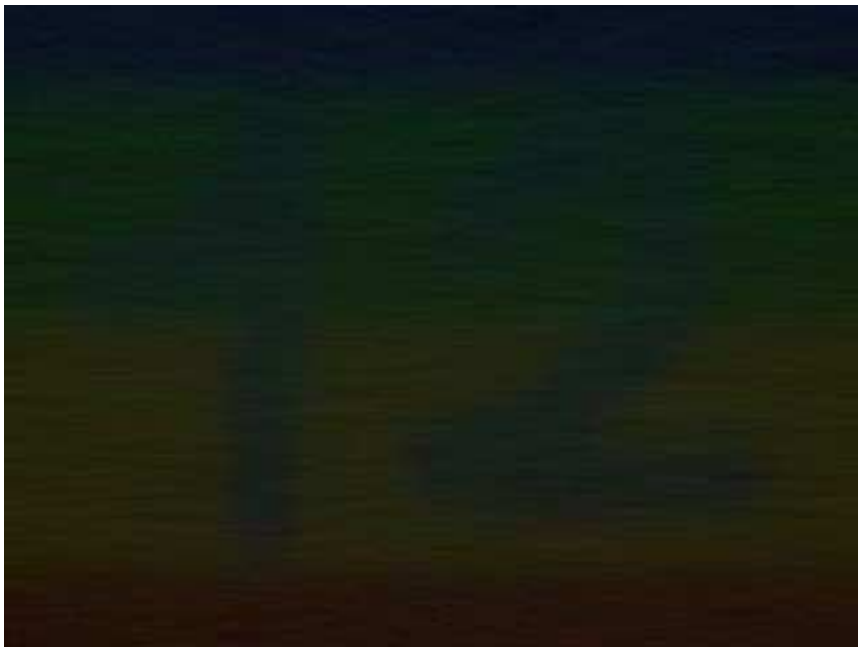


Obrázek 4.6: Obrázek získaný ještě před čočkou.

Obrázek 4.7: Obrázek získaný hned za čočkou.



Obrázek 4.8: Obrázek, na kterém už je částečně rozeznatelný text.

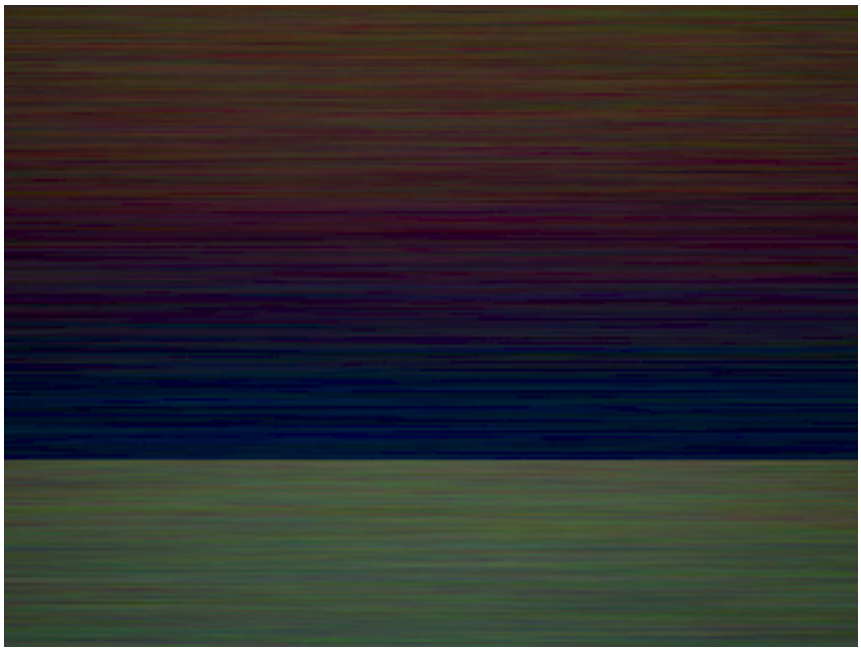


Obrázek 4.9: Nejostřejší získaný obrázek.

nekonečný hranol symetrický dle roviny kolmé na osu x. Zdroj i stínítko jsou umístěny do stejné vzdálenosti od hranolu. Po provedení simulace se na stínítku objeví obrázek 4.10. Spodní světlejší část obrázku zasahuje dovnitř hranolu, kde je mnohem více světla. Pak je vidět jasné rozhraní, kde stínítko opouští hranol. Od tohoto rozhraní nahoru je obrázek tmavý a až v horní části opět světlá. To je způsobeno tím, že světlo, které mělo dopadnout do tmavé části, bylo uvnitř hranolu odraženo zpět, místo aby prošlo do prostředí s nižším indexem lomu a poté bylo zachyceno stínítkem.

4.5 Zjišťování ohniskové vzdálenosti

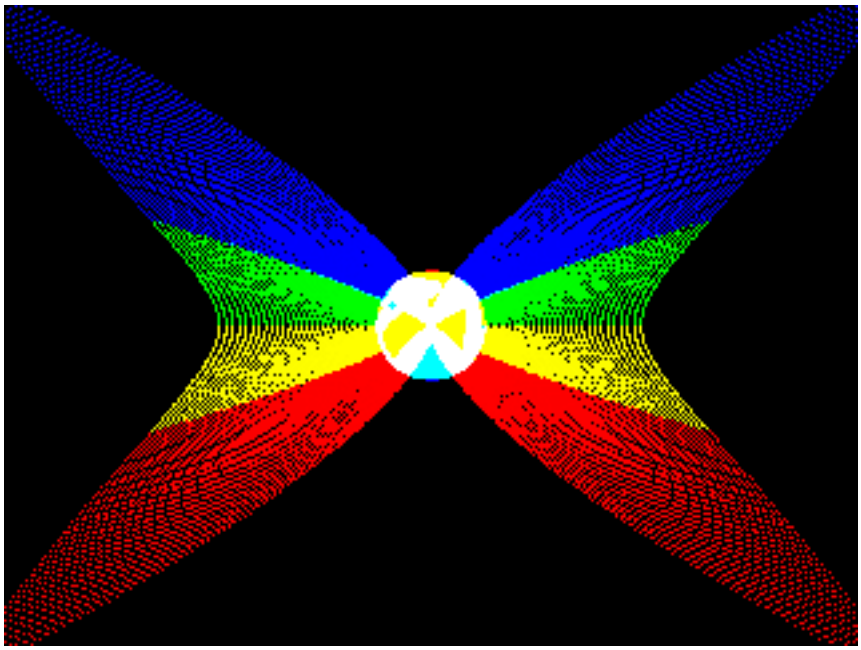
Pro zjišťování ohniskové vzálenosti použiji stejnou optickou soustavu, jako v druhém příkladě. Jediným rozdílem bude, že místo generování paprsků ze zdrojového obrázku do všech směrů s intenzitami podle Lambertova kosínového zákona budu generovat pouze paprsky kolmé. Tím že jsem použil více stínítek, mohu zjistit na kterém je výsledný obrázek nejmenší, což značí, že v tomto místě je svazek paprsků nejužší. Takto objevíme umístění ohniska, jelikož všechny paprsky ze svazku rovnoběžných paprsků dopadajících na



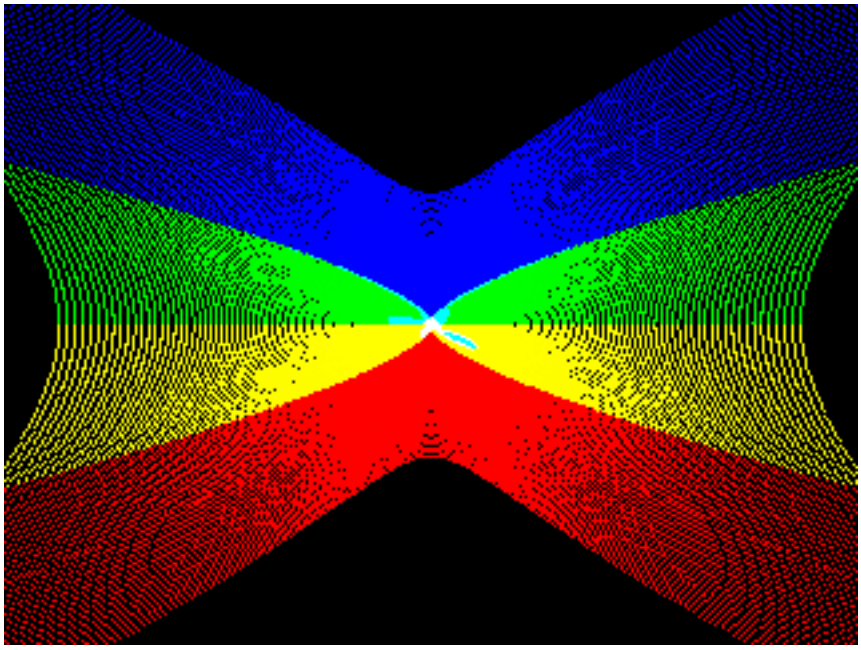
Obrázek 4.10: Obrázek světla, které prošlo skrz hranol.

spojku se lomí tak, že procházejí ohniskem. V tomto příkladě využiji schéma ze souboru `priklad4.xml`. Jako vstupní obrázek poslouží opět obrázek 4.5.

Z celé výstupní série jsem vybral jen obrázky, které jsou nejblíže ohnisku jsou to obrázek 4.11, který je těsně před ohniskem, a obrázek 4.12, který je těsně za ohniskem. První obrázek je ve vzdálenosti 9.6 jednotek od středu čočky, druhý ve vzdálenosti 9.8. Hledané ohnisko tedy leží někde v tomto intervalu, bude spíše blíže hodnotě 9.8, jelikož na druhém obrázku ještě je bílý kruh uprostřed obrázku, avšak je mnohem menší, než je tomu v případě obrázku prvního.



Obrázek 4.11: Poslední obrázek před ohniskem.



Obrázek 4.12: První obrázek za ohniskem.

Kapitola 5

Závěr

5.1 Shrnutí

V předkládané práci jsem představil implementaci programu pro simulování optických soustav. Popsal jsem funkce tohoto programu a jeho návrh.
Na předložené sadě příkladů jsem ukázal jeho vlastnosti, tedy to, jaké optické jevy dokáže simulovat. Při tvorbě příkladů jsem zjistil, že pro rozumné výsledné obrázky je třeba simulovat alespoň 300 papprsků z jednoho pixelu. Při počtu 1000 paprsků už jsou obrázky opravdu pěkné. Bohužel i při tomto množství použitých paprsků se ukazuje hlavní nevýhoda použité metody, kterou je vysoká zašuměnost výsledných obrázků. Ale i přes to lze z výsledných obrázků odhadnout parametry použité optické soustavy.

5.2 Možná rozšíření

Program OpticalBench lze ještě hodně rozšířit, mezi první možné rozšíření patří nahrazení konstanty pro množství odraženého světla funkcí dle Fresnelových vzorců. To by umožnilo simulovat odrazy uvnitř čoček. S použitím Fresnelových vzorců souvisí přidání možnosti simulovat polarizované světlo. Dalším rozšířením tedy je přidání polarizace pro využívané světlo. Možným rozřířením je i úprava získaných obrázků metodami používanými při zpracování fotonových map, tím by se zmenšilo zašumnění výsledku. V neposlední řadě lze přidat podporu pro další grafické rastrové formáty, především pro formáty využívající vyššího dynamického rozsahu jako například HDR.

Příloha A

Obsah CD-ROM

Součástí této práce je i přiložený CD-ROM, který obsahuje text práce, zdrojové soubory, několik vzorových schémat optických soustav, příklady vstupních a vygenerovaných obrázků. Adresářová struktura přiloženého CD-ROM:

- `dokumenty` – obsahují text práce a programátorskou dokumentaci ve formátu PDF.
- `instalace` – obsahuje instalátor programu OpticalBench a kopii složky s již nainstalovaným programem.
- `priklady` – obsahuje příklady použití programu uvedené v této práci
- `zdrojove_soubory` – obsahuje zdrojové soubory programu OpticalBench a projekt pro Microsoft Visual Studio.

Příloha B

Uživatelská dokumentace

B.1 Instalace

Program OpticalBench je napsán pro operační systém Windows. Instalaci provádí instalační soubor OpticalBenchsetup-1-0.exe. Při průběhu instalace můžete zvolit cílový adresář, kam se má program nainstalovat, jestli se má program přidat do seznamu v nabídce start, či jestli se má umístit zástupce na pracovní plochu.

B.2 Syntaxe paramentrů

Program OpticalBench se spouští z příkazové řádky následujícími způsoby:

```
./OpticalBench.exe -d davka
./OpticalBench.exe -s schema -p pocet [-k]
```

Význam jednotlivých parametrů je:

- `davka` – cesta k souboru obsahujícímu data pro dávkové zpracování
- `schema` – cesta k souboru obsahujícímu XML schéma optické soustavy
- `pocet` – celé číslo v rozsahu od 1 do 1000000, udává počet paprsků, které se budou generovat z 1 pixelu vstupního obrázku

Přepínač `-k` způsobí, že budou ze vstupních obrázků generovány pouze kolmé paprsky, toho se dá využít při zjišťování ohniskových vzdáleností optických soustav.

B.3 Definice schématu optické soustavy

Schéma optické soustavy je definováno v XML souboru. Jelikož optická soustava je v programu definována CSG stromem, členění XML schématu optické soustavy tomu odpovídá. Hlavním elementem je element `lavice`, což je kořen celé hierarchie. Pro kontrolu syntaktické správnosti schématu je k programu přiložen DTD dokument s názvem schema.dtd. Tento soubor je nutné přikopírovat k vaší definici schématu optické soustavy a na začátku schématu se na něj odkažte pomocí `<!DOCTYPE lavice SYSTEM "schema.dtd">`.

B.3.1 Použité elementy

Pro definování optické soustavy jsou používány následující elementy:

- `lavice` – je kořenem celého XML i definice optické soustavy, musí obsahovat elementy: `materialy`, `prostredi`, `cocky`, `zdroje`, `stinitka` a `clonky`.
- `materialy` – sdružuje v sobě definice jednotlivých materiálů, může obsahovat pouze jediný element, kterým je `material`, přičemž tento se může opakovat nebo se nemusí vyskytovat vůbec.
- `material` – definuje jednotlivý materiál, a obaluje vlastnosti, které materiál může mít, tento element se vyskytuje ve dvou případech. Prvním je definice nového materiálu, pak musí obsahovat elementy: `name`, právě jeden z dvojice `index` nebo `selmeier`, dále `odrazivost` a `popis`. Přičemž element index se použije pro materiál s konstantním indexem lomu a element selmeier pro materiál s indexem lomu závislým na vlnové délce. Obsah elementu `name` musí být pro každou definici nového materiálu unikátní, jelikož se materiály rozlišují dle jména. Druhým případem je odkaz na již definovaný materiál z elementu `len`, v takovém případě obsahuje pouze element `name`.
- `name` – definuje název, jeho obsahem je pouze řetězec znaků.
- `index` – definuje index lomu, jeho obsahem je desetinné číslo udávající index lomu světla daného materiálu.
- `odrazivost` – definuje odrazivost materiálu, udává kolik procent dopadajícího světla se odrazí. Jeho obsahem je desetinné číslo v rozmezí

od 0 do 1 přičemž 0 znamená, že se neodrazí žádné světlo a 1, že se odrazí veškeré světlo na tento materiál dopadající.

- `popis` – definuje popis materiálu, lze do něj uložit doplňující informace. Jeho obsahem je řetězec znaků.

- `cocky` – sdružuje v sobě definice jednotlivých čoček, případně jejich částí. Může zůstat prázdný, pokud soustava neobsahuje žádné čočky. Pokud soustava obsahuje alespoň jednu čočku, musí obsahovat alespoň jeden element `len` a může opakovaně obsahovat jeden z elementů: `union`,`intersection` nebo `difference`. Přičemž při každém opakování lze z této trojice zvolit jiný element.

- `len` – definuje čočku, může se stejně jako element `material` vyskytovat ve dvou případech. Prvním z nich je definice nové elementární části čočky. Z této definice při načítání schématu vznikne list CSG stromu. V tomto případě musí obsahovat elementy `name` a `shape`. V druhém případě slouží jako odkaz na již existující čočku, nebo část čočky. V tom případě musí obsahovat elementy `name` a `transformace`.

- `shape` – definuje základní těleso, musí obsahovat elementy `material`, `basicshape` a `transformace`. Přičemž mezi elementy `basicshape` a `transformace` lze vložit element `parametr`.

- `basicshape` – definuje elementární těleso. Jeho obsahem je řetězec znaků, přičemž tyto znaky určují název elementárního tělesa. Přípustné názvy těles jsou:

  - `Koule` pro jednotkovou kouli se středem v počátku soustavy souřadnic.
  - `Krychle` pro jednotkovou krychli se středem v počátku soustavy souřadnic.
  - `Poloprostor` pro poloprostor, který má hraniční rovinu `x=0` a objem má směrem k záporné části osy x.
  - `Valec` pro nekonečný válec o poloměru 1 jednotka, jehož osou je osa x soustavy souřadnic.
  - `Kuzel` pro dvojitý nekonečný kužel s vrcholem v počátku soustavy souřadnic, jehož osou je osa x soustavy souřadnic.

  - `RotacniParaboloid` pro rotační paraboloid s vrcholem v počátku soustavy souřadnic a s objemem směřujícím v kladném směru osy x.
  - `RotacniHyperboloid` pro rotační hyperboloid, jehož osou je osa x oustavy souřadnic. Podle obsahu elementu `parametr` v nadřazeném elementu `shape` se rozlišuje, jestli jde o dvojdílný nebo jednodílný hyperboloid. Pokud je jeho obsahem řetězec: „Jednodilny“, pak se jedná o jednodílný hyperboloid, v ostatních případech jde o hyperboloid dvojdílný.

- `parametr` – definuje, jestli se použitý rotační hyperboloid vyskytuje ve verzi jednodílné nebo dvojdílné. To je rozlišeno tím, jestli je obsahem tohoto elementu „Jednodilny“ nebo něco jiného.

- `transformace` – definuje transformační matici, kterou se daný objekt transformuje. Obsahuje položky `scale`, `rotateX`, `rotateY`, `rotateZ`, `translate` a `matrix`. Tyto položky se za sebou mohou libovolně opakovat, přičemž položky se načítají v tom pořadí v jakém jsou definovány.

- `scale` – definuje transformační matici pro zvětšení. Obsahuje tři desetinná čísla oddělená mezerami. První číslo udává zvětšení ve směru osy x, druhé ve směru osy y a třetí ve směru osy z. Čísla nesmí být rovná nule.

- `rotateX` – definuje transformační matici pro otočení kolem osy x. Obsahuje jedno desetinné číslo, které udává úhel pootočení v radiánech. Úhel stoupá ve směru od osy y k ose z.

- `rotateY` – definuje transformační matici pro otočení kolem osy y. Obsahuje jedno desetinné číslo, které udává úhel pootočení v radiánech. Úhel stoupá ve směru od osy x k ose z.

- `rotateZ` – definuje transformační matici pro otočení kolem osy z. Obsahuje jedno desetinné číslo, které udává úhel pootočení v radiánech. Úhel stoupá ve směru od osy x k ose y.

- `translate` – definuje transformační matici pro posunutí. Obsahuje tři desetiná čísla oddělená mezerami. První udává posun ve směru osy x, druhé ve směru osy y a třetí ve směru osy z. Kladné číslo značí posun v kladném směru, záporné číslo posun v záporném směru.

- `union` – definuje sjednocení dvou čoček, musí obsahovat elementy `name`,`len` a `len`, kde name bude jméno takto vzniklé čočky a len jsou odkazy na již definované čočky, respektive jejich názvy.

- `intersection` – definuje průnik dvou čoček, musí obsahovat elementy `name`,`len` a `len`, kde name bude jméno takto vzniklé čočky a len jsou odkazy na již definované čočky, respektive jejich názvy.

- `difference` – definuje rozdíl dvou čoček, musí obsahovat elementy `name`,`len` a `len`, kde name bude jméno takto vzniklé čočky a len jsou odkazy na již definované čočky, respektive jejich názvy. Odečítá se druhá zde použitá od první.

- `zdroje` – sdružuje v sobě všechny zdroje, jimiž jsou zářící textury. Obsahuje alespoň jeden element `zdroj`, ale může jich obsahovat i více.

- `zdroj` – reprezentuje jednu vstupní rovinu s odkazem na obrázek, který se z této roviny promítá. Obsahuje elementy: `obrazek`, `soubor`, `pxNj`, `typ`, `rovina`. Z nichž element `obrazek` nemusí být přítomen.

- `soubor`– definuje název souboru, který musí být formátu TGA. Obrázek uložený v tomto souboru bude připojen k vstupní(generující) rovině a do souboru s tímto názvem bude uložen obrázek získaný na výstupní rovině.

- `pxNj` – definuje, poměr mezi pixely obrázku a jednotkami používanými v optické soustavě. Hodnotou je kladné desetinné číslo větší než 0.

- `typ` – definuje, o jaký typ vstupní(generující), případně výstupní roviny se jedná. Obsahuje textový řetězec, který může nabývat pouze hodnot: „virtualni“ nebo „skutecna“. Přičemž první znamená, že po dopadu paprsku na danou výstupní rovinu se pouze zanesou jeho informace do příslušného obrázku a paprsek není nijak ovlivněn. V druhém případě se též zanesou informace, ale paprsek je touto rovinou pohlcen a už dále nepokračuje. V případě dopadu na vstupní rovinu se paprsek chová stejně, jen se nezaznamenávají žádné informace.

- `rovina` – definuje umístění roviny se vstupním, nebo výstupním obrázkem. Jde o rovinu `x=0` v systému souřadnic. Obrázky se na tuto rovinu mapují tak, že střed obrázku je protnut osou x systému souřadnic. Obrázky se na rovinu mapují z kladné strany osy x. Přičemž levý

horní roh obrázku má kladné souřadnice na osách y a z. Při generování paprsků z této roviny, jsou paprsky vysílány v kladném směru osy x. Při přijímáni jsou přijímany pouze paprsky se směrem opačným než je kladný směr osy x.

- `stinitka` – sdružuje v sobě všechna stínítka, pomocí nich se získávají výsledné obrázky. Obsahuje alespoň jeden element stínítko, ale může jich obsahovat i více.
- `stinitko` – reprezentuje jednu výstupní rovinu s odkazem na obrázek, do kterého se ukládají informace o paprscích dopadlých na tuto rovinu. Obsahuje elementy: `obrazek`, `soubor`, `pxNj`, `typ`, `rovina`.
- `sirka` – definuje šířku výstupního obrázku v pixelech, potažmo tak definuje i šířku stínítka. Hodnotou je kladné celé číslo větší než 10.
- `vyska` – definuje výšku výstupního obrázku v pixelech, potažmo tak definuje i výšku stínítka. Hodnotou je kladné celé číslo větší než 10.
- `clonky` – sdružuje v sobě všechny clonky, připadně jejich části. Může zůstat prázdný pokud soustava neobsahuje čádné clonky. Je to ekvivalent elementu `len` pro cocky. Pokud soustava obsahuje alespoň jednu clonku, musí obsahovat alespoň jeden element `len` a může opakovaně obsahovat jeden z elementů: `union`,`intersection` nebo `difference`. Přičemž při každém opakování lze z této trojice zvolit jiný element.
- `obrazek` – při použití u stínítka definuje, jakého typu je obrázek spojený se stínítkem. Možné hodnoty jsou:
  - `GreyScale` – udává, že daný obrázek bude uložen jen v odstínech šedi na každý pixel připadá 1 byte.
  - `RGB` – udává, že daný obrázek bude uložen ve formátu RGB, kde na každou barvu připadá 1 byte.
  - `cislo` – `cislo` je celočíselná hodnota od 4 do 340 a udává na kolik částí má být rozděleno spektrum při zaznamenávání hodnot do obrázku. Při uložení se toto spektrum převede do formátu RGB jako výše.
- `prostredi` – definuje prostředí v okolí optické soustavy. Hodnotou je řetězec znaků, který udává název nějakého materiálu definovaného v elementu `materialy`.

B.4 Dávkové zpracování

Program umožňuje dávkové zpracování více předpřipravených soustav. Příkazy pro dávkové zpracování se zapisují do jednoduchého textového souboru. Příkaz pro zpracování jedné dávky se skládá ze čtyř částí:

- název schématu s definicí optické soustavy
- počet paprsků
- jména vstupních obrázků
- jména výstupních obrázků

Každá z těchto částí se nachází na samostatném řádku. V druhém a dalších příkazech lze vynechat název schématu optické soustavy, v takovém případě se použije soustava z posledního předchozího příkazu, ve kterém byl název soustavy definován.

Počet paprsů je celé číslo v rozsahu od 1 do 1000000, které udává počet paprsků vyslaných z jednoho pixelu vstupního obrázku. Za tímto číslem může být ještě písmeno `k`, což značí, že při generování paprsků z obrázků budou generovány pouze kolmé paprsky. Jména vstupních a výstupních obrázků jsou cesty k souborům formátu TGA oddělené mezerami. Mezi každými dvěma cestami je právě jedna mezera.
Počet vstupních obrázků se musí schodovat s počtem zdrojů, definovaných ve schématu optické soustavy. Stejně tak počet výstupních obrázků se musí schodovat s počtem stínítek. Pokud se jeden z těchto počtů neschoduje, celý příkaz pro dávkové zpracování se přeskočí a pokračuje se dalším příkazem.

Literatura

[1] Glassner A. S.: *Principles of digital image synthesis. Volume one.*, Morgan Kaufmann Publishers, San Francisco, 1995, ISBN 1-55860-276-3

[2] Glassner A. S.: *Principles of digital image synthesis. Volume two.*, Morgan Kaufmann Publishers, San Francisco, 1995, ISBN 1-55860-276-3

[3] Žára J., Beneš B., Sochor J., Felkel P.: *Moderní počítačová grafika*, Computer Press, Praha, 2004, ISBN 80-251-0454-0

[4] Shirley P., Morley R. K.: *Realistic Ray Tracing*, A K Peters, Natick, 2003, ISBN 1-56881-198-5

[5] Pelikán J.: *Fyzikální světelné modely*,[online]. Fyzikální světelné modely [cit. 2010-07-15]. Dostupné na internetu:
`http://www.studopory.vsb.cz/studijnimaterialy/Sbirka_Fyzika/`

[6] Plášek J., Procházka M., Antoš R.: *Optika pro počítačovou grafiku* [online]. infooptika01.pdf [cit. 2010-07-10]. Dostupné na internetu:
`http://is.cuni.cz/studium/predmety/index.php?do=predmet&kod=NPGR030`

[7] Barčová K., Janurová E., Kopečná M., Kušnerová M., Uhlář R.: *Sbírka úloh z fyziky*,[online]. Geometrická optika (4.2) [cit. 2010-07-12]. Dostupné na internetu:
`http://www.studopory.vsb.cz/studijnimaterialy/Sbirka_Fyzika/`