VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

FOTOREALISTICKÉ ZOBRAZOVÁNÍ

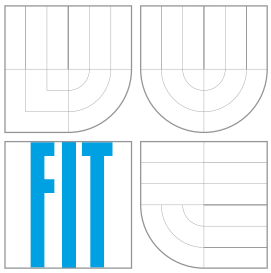
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ LYSEK

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

FOTOREALISTICKÉ ZOBRAZOVÁNÍ

PHOTOREALISTIC RENDERING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ LYSEK

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Dr. Ing.. PAVEL ZEMČÍK

BRNO 2013

Abstrakt

Tato práce obsahuje popis implementace fotorealistické metody zobrazení scény. V první části práce jsou popsány postupy a techniky, které slouží k fotorealistickému zobrazení scény. Jsou zde uvedeny jak metody zobrazení scény (raytracing, metoda radiozity nebo photon mapping) tak další techniky sloužící v počítačové grafice. V druhé části práce je zhodnocený současný stav, uveden důvod výběru tématu a detailněji specifikováno zadaní. Poslední část práce popisuje postup implementování fotorealistické metody zobrazení.

Abstract

The thesis describes implementation of photorealistic method. The first part describes the procedures and techniques, that are used to display photorealistic scenes such as raytracing, radiosity or photon mapping and other techniques used in computer graphics. The second part of thesis is focused on evaluation of current state of problem. In this part is also described reason of choosing this topic and thesis assignment is there more specified. The last part of thesis is focused on implementation details of photorealistic method.

Klíčová slova

metoda sledovaní paprsku, anti aliasing, rozdělení podprostoru, oktalový strom, fotorealistické zobrazení, raycasting, radiozita, photon mapping, phongův osvětlovací model, renderovací rovnice, dvousměrná odrazová distribuční funkce

Keywords

raytracing, antialiasing, spatial subdivision, octree, photorealistic rendering, raycasting, radisity, photong mapping, phong lighting, rendering equation, bidirectional reflectance distribution function

Citace

Tomáš Lysek: Fotorealistické zobrazování, bakalářská práce, Brno, FIT VUT v Brně, 2013

Fotorealistické zobrazování

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Prof. Dr. Ing. Pavla Zemčíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Lysek
13. května 2013

Poděkování

Děkuji vedoucímu práce Prof. Dr. Ing. Pavlovi Zemčíkovi za odborné vedení a velmi cenné rady, které vedly k úspěšnému dokončení této práce.

Obsah

Kapitola 1

Úvod

Počítačová grafika se v poslední době stala velmi důležitým odvětvím informatiky, její produkty nás potkávají, buď přímo, nebo nepřímo, na každém kroku, ať už jde o strojírenství, architekturu, návrh designu, nebo ve filmovém a herním průmyslu. Takový rozmach počítačové grafiky zajistil hlavně fakt, že její produkty se co nejvíce podobají reálným snímkům a někdy bývají levnější anebo i jedinou možností k pořízení daného snímku (např. filmy ve vesmíru).

Již od začátku vývoje počítačové grafiky, byla snaha o co největší realističnost výsledného obrazu. V dnešní době je velký tlak na co nejrealističtější grafické výsledky, aby výsledný produkt byl co nejvíce věrný reálnému světu (filmy, hry). V poslední době byl v této oblasti pokrok poměrně výrazný, pro demonstraci je možné vzít příklad z filmového průmyslu a porovnat fotorealističnost filmů z dnešní doby a z doby 20 (ale i 10) let nazpět. Dá se říct, že pokrok ve fotorealistické počítačové grafice, kdy se budou nacházet lepší a optimalizovanější metody zobrazení, bude s počítačovou grafikou vždy úzce spjat.

Cílem této bakalářské práce je hlavně snaha o hlubší seznámení s počítačovou grafikou a naprogramování softwaru pro zobrazení scény pomocí fotorealistické metody.

V první části práce jsou popsány postupy a techniky, které slouží k fotorealistickému zobrazení scény. Jsou zde uvedeny jak metody zobrazení scény (raytracing, metoda radiozity nebo photon mapping) tak další techniky sloužící v počítačové grafice. V druhé části práce je zhodnocený současný stav, uveden důvod výběru tématu a detailněji popsáno zadaní. Poslední část práce popisuje postup implementování fotorealistické metody zobrazení.

Kapitola 2

Fotorealistické zobrazovaní

Tato kapitola popisuje principy a postupy počítačové grafiky, které slouží pro zobrazení fotorealistické scény. Tato kapitola si neklade za cíl encyklopedicky popsat vše co se týká počítačové grafiky a fotorealistického zobrazení. Popsání tohoto problému by bylo poměrně značně obsáhlé, proto je v této části práce popsáno pouze to, z čeho autor práce vycházel při tvorbě práce.

2.1 Fotorealistické zobrazovaní

Zobrazování (nebo anglicky rendering) je proces, při kterém se generuje dvourozměrný obraz z trojrozměrné scény [12]. Fotorealistické zobrazování je zobrazování, kdy vygenerovaný obraz co nejvíce odpovídá skutečnosti - realitě.

V počítačové grafice bývá hraniční reprezentace objektů modelu scény nejčastěji reprezentována jako síť trojúhelníků, kdy v každém z vrcholů trojúhelníku jsou uloženy normály povrchu. Normály povrchu slouží k výpočtu osvětlovacího modelu - barvy objektu v daném bodě. Pro určení barvy objektu, je nutné mít pro každý objekt scény, uloženy informace o materiálu objektu - jako barva, zář, odrazivost, průhlednost atd. Více o výpočtu lokálního osvětlovacího modelu pojednává kapitola 2.4.

V počítačové grafice existují nefotorealistické metody zobrazení, které se snaží, na úkor kvality zobrazení, zobrazit scénu v reálném čase.

Fotorealistické metody si nekladou za důraz zobrazit scénu co nejrychleji, ale v co největší kvalitě, takže ve výsledných obrazech jsou prvky jako odrazy nebo lomy paprsků, stíny.

2.2 Promítání

Promítání (neboli projekce) je transformace trojrozměrné scény do dvourozměrného obrazu. V počítačové grafice se nejčastější promítá pomocí perspektivního promítání, kdy všechny *promítací paprsky* směřují do jednoho bodu - *zobrazovacího bodu*, který se dá představit jako kamera (nebo oko). Zobrazovací paprsky procházejí přes průmětnu (*viewing plane*) – plochu v prostoru – a v místě dopadu vytvářejí průmět (obraz na průmětně)[12].

2.3 Zobrazování scény rasterizací trojúhelníku

Zobrazování scény rasterizací trojúhelníku je jedna z nejvyužívanějších zobrazovacích metod, protože tuto metoda je možno hardwarově akcelerovat. Použití této metody a hardwarové akcelerace se docílí vykreslení scény v reálném čase.

V prvním kroku se scéna transformuje tak, aby průmětna a zobrazovací bod ležely na ose z. Ze scény jsou poté ořezány objekty pomocí pohledového objemu, kdy se ze zobrazované scény odstraní objekty, které se nemohou zobrazit na průmětně. Pohledový objem je většinou jehlan, kdy vrchol jehlanu je zobrazovací bod.

V takto ořezané scéně se z každého trojúhelníku vezmou jeho vrcholy a pomocí transformační matice pro perspektivní projekci se vypočítají souřadnice vrcholů trojúhelníku na průmětně. Trojúhelník je poté, například pomocí Pinedova rasterizačního algoritmu, vykreslen na průmětně/framebufferu.

Barva jednoho pixelu zobrazovaného trojúhelníku se určí pomocí výpočtu lokálního osvětlovacího modelu popsaného v kapitole 2.4. Způsob jakým se dopočítává barva vnitřních bodů trojúhelníku se určí pomocí metod stínování popsaných v kapitole 2.5.

Před vykreslením vypočítaného pixelu do framebufferu, je nutné určit jeho viditelnost. Pro určení viditelnosti je jeden z nejpoužívanějších algoritmu algortimus s názvem z-buffer. Díky tomu, že byla scéna v prvním kroku přetransformována tak, aby zobrazovací bod a průmětna ležely na ose z, je možné vzdálenost od průmětny určit podle vzdálenosti bodu na ose z. Při vykreslování je vytvořena pomocná struktura o stejné velikosti jako je framebuffer, před zapsáním pixelu do framebufferu se nejprve zkontroluje, zda-li na stejné pozici není vykreslen pixel s menší hodnotou souřadnice z, pokud není, bod se uloží do framebufferu a do z-bufferu se uloží souřadnice z zobrazovaného bodu.

Hlavní výhodou této metody je její velká rychlost, dosažena hlavně díky rasterizaci objektů scény na dvourozměrné průmětně a díky možností paralelizace této metody. Problém této metody je v její nefotorealističnosti. V základní podobě chybí ve scéně stíny (které se musí pomocí určitých algoritmů dopočítávat) a je nutné řešit viditelnosti vykreslovaných pixelů.

2.4 Lokální osvětlovací model

Tato podkapitola se bude věnovat výpočtu barvy pomocí lokálního osvětlovacího modelu, výpočtu barvy jediného bodu na povrchu daného objektu.

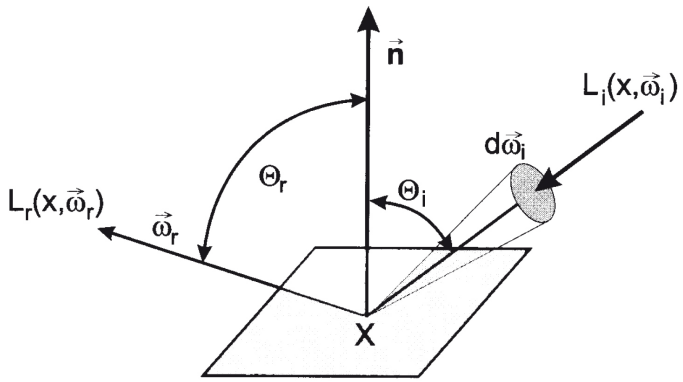
První část této kapitoly se bude zabývat dvousměrovou odrazovou distribuční funkcí (BRDF).

Druhá část se bude zabývat empirickými osvětlovacími modely.

Dvousměrná odrazová distribuční funkce - BRDF

V reálném světě světlo, které vnímáme, je světlo, které se odráží od povrchu nějakých objektů. Barva odraženého světla je dána spektrální charakteristikou dopadajícího světla a především vlastnosti povrchu - směr odraženého světla a určité vlnové délky, který daný povrch odráží/pohlcuje.

Dvousměrná odrazová distribuční funkce - BRDF (z anglického Bidirectional Reflectance Distribution Function), kterou popsal Fred Nicodemus okolo roku 1965 v [14], charakterizuje odrazové schopnosti povrchu materiálu v určitém bodě [12].



Obrázek 2.1: Dvousměrná odrazová distribuční funkce - BRDF [12]

Obrázek 2.1 znázorňuje dvousměrovou odrazovou distribuční funkci. Mějme světlo dopadající ze směru $\vec{\omega_i}$ odražené směrem $\vec{\omega_r}$ v bodě x. BRDF se označuje jako $f_r(x, \vec{\omega_r}, \vec{\omega_i})$ a definuje poměr radiance odražené v daném bodě, označené jako $dL_r(x, \vec{\omega_r})$, ke vstupní diferenciální radianci $dL_i(x, \vec{\omega_i})$ promítnuté na kolmou plochu [12].

$$f_r(x, \vec{\omega_r}, \vec{\omega_i}) = \frac{dL_r(x, \vec{\omega_r})}{dL_i(x, \vec{\omega_i})(\vec{\omega_i}.\vec{n})d\vec{\omega_i}} \tag{2.1}$$

Rovnice 2.1 popisuje dvousměrovou odrazovou distribuční funkci BRDF. Z dané rovnice je patrné, že pokud světlo dopadá kolmo na plochu, přijatý výkon je největší.

Dvousměrová odrazová distribuční funkce - BRDF - má určité vlastnosti:

- *Pozitivita* - funkce BRDF není nikdy záporná $f_r(x, \vec{\omega_r}, \vec{\omega_i}) \geq 0$
- *Linearita* - hodnota BRDF pro určitý vstupní úhel $\vec{\omega_i}$ není závislá na hodnotách BRDF pro jiné vstupní úhly. Tato vlastnost BRDF znamená, že paprsek z daného směru je vyzářen bez ohledu na to, co přichází z okolních směrů.
- *Helmholtzův princip reciprocity* - hodnota BRDF v určitém bodě je stejná, když zaměníme směr dopadu za směr odrazu a obraceně, tedy $f_r(x, \vec{\omega_r}, \vec{\omega_i}) = f_r(x, \vec{\omega_i}, \vec{\omega_r})$
- *Anizotropie* - vlastnost materiálu, kdy odraz světla nezáleží na vstupním/výstupním směru a bodě x, ale taky natočení povrchu v bodě x kolem normálového vektoru k tomuto povrchu. Příkladem anizotropního materiálu je třeba hrubě nabroušený kov. Pro zobrazení této vlastnosti materiálu pomocí BRDF, je nutné zahrnout do BRDF funkce úhel ϕ, který určuje úhel natočení bodu k normále. V počítačové grafice se většinou s anizotropním materiálem nepracuje, proto se používá definice BRDF uvedená v rovnici 2.1.

Lokální osvětlovací model

Lokální osvětlovací model je model pomocí kterého se vypočítává barva jediného bodu na povrchu určitého objektu. Rovnice pro výpočet lokálního osvětlovacího modelu obsahuje dvousměrovou odrazovou distribuční funkci a vyjadřuje radianci $L_r(x_i, \vec{\omega_r})$, která se odrazí v daném směru pro všechny vstupní směry $\vec{\omega_i}$.

$$L_r(x, \vec{\omega_r}) = \int_{\Omega} f(x, \vec{\omega_r}, \vec{\omega_i}) L_i(x, \vec{\omega_i}) \cos\Theta \, \mathrm{d}\vec{\omega_i} \tag{2.2}$$

Tato rovnice udává celkovou radianci, která se odráží od povrchu v bodě x ve směru $\vec{\omega_r}$ a je spočtena jako integrál všech radiancí, které dopadají na povrch v bodě x vynásobené dvousměrovou odrazovou distribuční funkcí. Barva bodu na povrchu tělesa se tedy získá jako odražená radiance v daném bodě [12]. Z lokálního osvětlovacího modelu vychází empirické osvětlovací modely například: Phongův, Blinn-Phongův nebo Lambertův osvětlovací model.

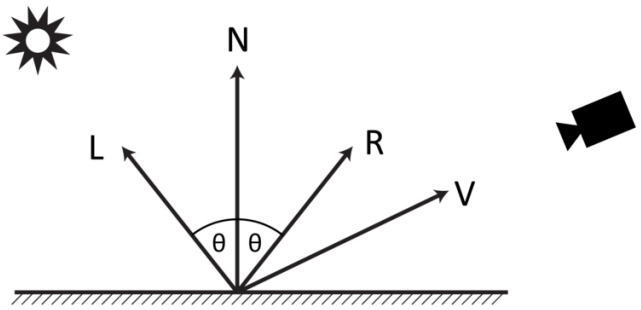
Phongův osvětlovací model

Phongův osvětlovací model je empirický osvětlovací model, který se požívá pro výpočet odraženého světla[12]. Byl představen vědcem Bui Tuong Phongem na univerzitě v Utahu v rámci jeho dizertační práce [15].

Pro tvorbu realistického vzhledu výsledného obrazu Phong určil, že se světlo v daném bodě skládá ze tří složek: okolní, difúzní a zrcadlové složky světla.

Pro výpočet Phongova osvětlovacího modelu se používají vektory:

- V - vektor udávající směr od bodu k pozorovateli - vektor pohledu
- N - normálový vektor plochy daného objektu
- L - vektor od bodu ke světelnému zdroji
- R - ideální odraz světelného vektoru L s povrchem objektu v daném bodě



Obrázek 2.2: Vektory použité pro vyhodnocení Phongova osvětlovacího modelu[22].

Na obrázku 2.2 jsou znázorněny vektory, které se používají pro vyhodnocení Phongova osvětlovacího modelu. Vektory N a L se používají pro výpočet difúzní složky světla. Vektory R a V se používají pro výpočet zrcadlové složky světla.

Okolní složka světla I_a

Okolní složka světla (anglicky Ambient light - někdy počeštěno na ambientní složka) je světlo, které dopadá rovnoměrně ze všech směrů. Napodobuje odražené světlo, které vzniklo mnohonásobnými odrazy od světelných zdrojů. Tato část výsledné barvy zajišťuje částečné osvětlení odvrácených ploch, nebo ploch, které jsou ve stínu se světelným zdrojem.

Příspěvek ambientního světla je určen vztahem:

$$I_a = I_A r_a \tag{2.3}$$

kde I_a je intenzita okolního světla ve scéně a $r_a \in (0, 1)$ je odrazový koeficient materiálu objektu, který vyjadřuje schopnost materiálu odrážet ambientní světlo. Intenzita okolního světla bývá ve většině scén konstantní pro celou scénu. Odrazový koeficient bývá totožný s koeficientem r_d u výpočtu difúzního světla (níže).

Difúzní složka světla I_d

Difúzní složka (diffuse light) světla I_d je definovaná vztahem:

$$I_d = I_L r_d(\vec{L}.\vec{N}) \tag{2.4}$$

kde I_L je barva dopadajícího světla, r_d je koeficient difúzního odrazu. Udává zastoupení difúzní složky v odraženém světle směrem k pozorovateli. Velikost intenzity světla I_d je větší podle toho čím víc je směr dopadu světla (vektoru $\vec{L}$) blíž normále $\vec{N}$. Z tohoto tvrzení se dá vyvodit, že difúzní složka světla zobrazuje trojrozměrný vzhled výsledného objektu. Difúzní složka se započítá do osvětlovacího modelu pouze v případě, kdy je vektorový součin $(\vec{L}.\vec{N})$ vetší než 0.

Zrcadlová složka I_s

Zrcadlová složka (specular light) je vyjádřena vztahem:

$$I_s = I_L r_s(\vec{V}.\vec{R})^h \tag{2.5}$$

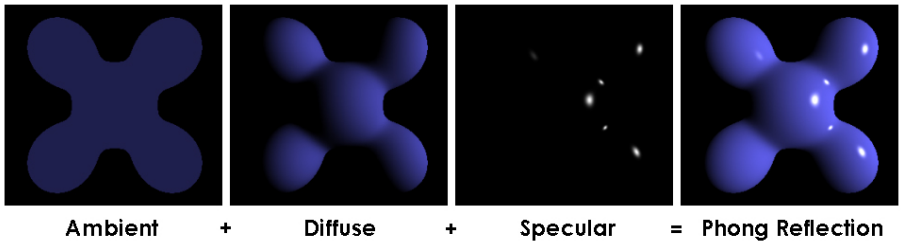
kde I_L je barva dopadajícího světla, $0 \geq r_s \geq 1$ je koeficient zrcadlového odrazu a určuje míru zastoupení zrcadlové složky ve výsledném odraženém odrazu. $\vec{V}$ je vektor pohledu. Vektor $\vec{R}$ je vektor vyjadřující směr ideálního zrcadlového odrazu a je symetrický s vektorem $\vec{L}$ podle normály $\vec{N}$. Dá se vypočítat vztahem $\vec{R} = 2(\vec{L}.\vec{N})\vec{N} - \vec{L}$. Z výše uvedeného vzorce vyplývá, že čím blíže je odražený vektor $\vec{R}$ blíž vektoru pohledu $\vec{V}$ tím světlejší bude výsledný příspěvek zrcadlové složky světla a tím více bude výsledná barva lesklejší.

Koeficient $h \in (1, \infty)$ vyjadřuje ostrost zrcadlového odrazu (Phong exponent). Čím větší bude koeficient h, tím je lesklá oblast na povrchu objektu menší, ale intenzivnější. Zrcadlová složka se započítá do osvětlovacího modelu pouze v případě, kdy je vektorový součin $(\vec{V}.\vec{R})$ vetší než 0.

Výsledné světlo

Po zjištění zrcadlové, difúzní a ambientní složky se celkové světlo I odražené od povrchu směrem k pozorovateli vypočítá jako:

$$I = I_s + I_d + I_a = I_L r_s (\vec{V}.\vec{R})^h + I_L r_d (\vec{L}.\vec{N}) + I_A r_a \tag{2.6}$$



Obrázek 2.3: Dílčí světelné složky[22].

Na obrázku 2.3 jsou znázorněny dílčí světelné složky Phongova osvětlovacího modelu a konečný výsledek vyhodnocení Phongova osvětlovacího modelu. Phongův osvětlovací model není fyzikálně korektní, protože nesplňuje vlastnosti dvousměrové distribuční funkce BRDF, ambientní složka světla je počítána jako konstanta pro celý objekt (scénu), přitom ve fyzikálně korektním modelu má být intenzita světla rovna součtu všech dopadajících paprsků na daný bod. Existují však určité adaptace Phongova osvětlovacího modelu, které jsou fyzikálně korektní například popsané v [2].

Lambertův osvětlovací model

Lambertův osvětlovací model se používá pro osvětlení matných objektů, kdy materiál objektu odrazí světlo do všech směrů se stejnou intenzitou [5]. Výpočet tohoto osvětlovacího modelu je stejný jako výpočet difúzní složky světla u phongova modelu:

$$I = I_L r_d (\vec{L}.\vec{N}) \tag{2.7}$$

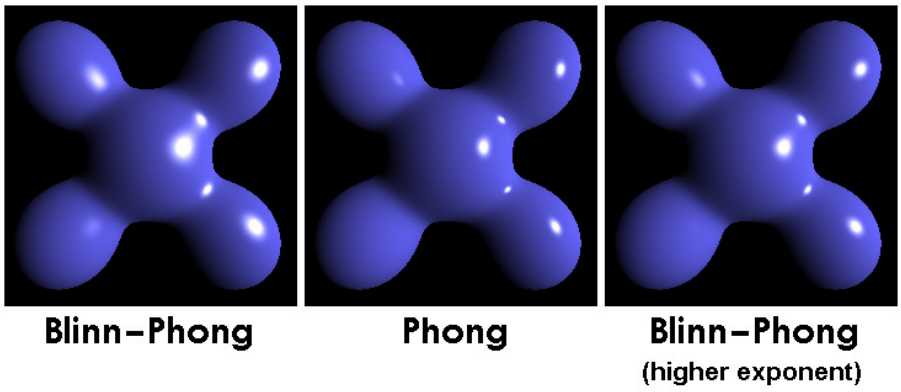
Hodnota osvětlení je tedy závislá pouze na úhlu mezi vektorem $\vec{L}$, označující světelný paprsek a normálou N.

Blinn-Phongův osvětlovací model

Blinn-Phongův osvětlovací model je modifikací Phongova osvětlovacího modelu. Modifikace spočívá při výpočtu zrcadlové složky, kdy se nahradí vektorový součin odraženého paprsku světla $\vec{R}$ s vektorem k pozorovateli $\vec{V}$ za vektorový součin půlvektoru $\vec{H}$ s normálou povrchu $\vec{N}$. Půlvektor se vypočítá jako:

$$\vec{H} = \frac{\vec{L} + \vec{V}}{\|\vec{L} + \vec{V}\|} \tag{2.8}$$

nahrazení vektorového součtu $\vec{R}.\vec{V}$ za $\vec{H}.\vec{N}$ se děje z toho důvodu, že výpočet odraženého vektoru světla R je časově náročnější než výpočet půlvektoru $\vec{H}$.



Obrázek 2.4: Rozdíl mezi Blin-Phongovým a Phongovým osvětlovacím modelem[21].

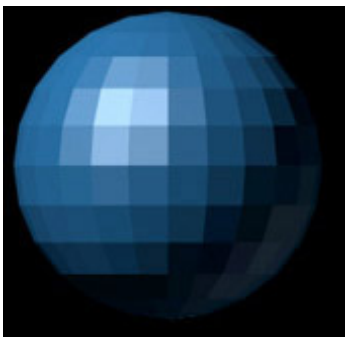
Na obrázku 2.4 je znázorněn rozdíl mezi Blin-Phongovým a Phongovým osvětlovacím modelem, z obrázku je patrné, že oba modely dělí jen malý rozdíl. Z důvodu rychlejšího výpočtu se Blinn-Phongův model používá v OpenGL a DirectX.

2.5 Metody stínování

Metody stínování se zabývají urychlením výpočtu při vyhodnocování povrchu objektů ve scéně. Počítání osvětlovacího modelu pro každý bod ve scéně je pro použití v reálném čase zbytečně zdlouhavé. Proto byly vytvořeny metody *stínování* (anglicky shading). Pomocí stínování se zobrazují křivosti a zaoblení ploch a vytváří se prostorový obraz scény [12].

Konstantní stínování

Pod pojmem konstantní stínování (anglicky flat shading) se skrývá metoda, která předpokládá jedinou normálu pro každou plochu objektu a používá se pro zobrazení rovinných těles [12]. Pokud se předpokládá jediná normála pro celou plochu, je jednoduché vypočítat pomocí osvětlovacího modelu barvu v daném bodě a tuto barvu použít pro celou tuto plochu.

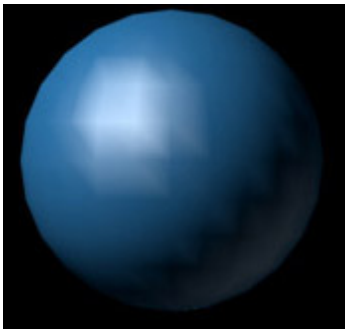


Obrázek 2.5: Scéna s koulí vykreslená pomocí konstantního stínování[16].

Metoda konstantního stínování je velmi rychlá, ale pro její značnou nerealističnost, kdy lze vidět složení modelu z plošek u oblých modelů (z důvodu konstantní barvy po celém obsahu plošky), se již nepoužívá.

Gouraudovo stínování

Gouraudovo stínování bylo poprvé představeno Henri Gouraudem v roce 1971 [8]. Tato metoda je vhodná pro plynulé stínování zakřivených těles (koule, válec, ...) tvořených množinou rovinných ploch. Podstata této metody spočívá ve vypočítání osvětlovacího modelu ve vrcholech vykreslované plochy následované dopočítáním barev vnitřních bodů pomocí bilineární interpolace.

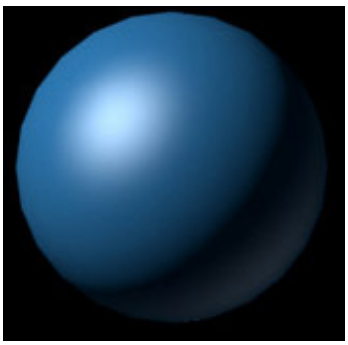


Obrázek 2.6: Scéna s koulí vykreslená pomocí Gouraudova stínování[16].

Tato metoda pracuje pouze s ambientní a difúzní složkou světla. Zobrazené výsledky jsou lepší než výsledky vypočítané pomocí konstantního stínování. Kvalita zobrazení je závislá na jemnosti dělení objektu do roviných ploch - čím jemnější dělení, tím kvalitnější výsledek za cenu větší výpočetní náročnosti.

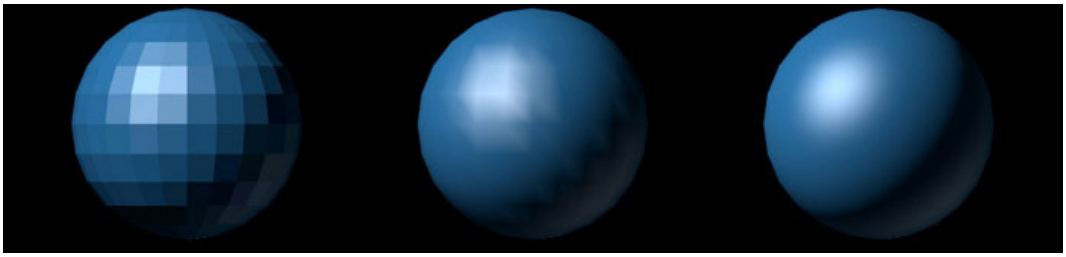
Phongovo stínováni

Poslední ze tří uvedených metod je phongovo stínování (anglicky Phong shading). Tato metoda stínování byla představena Bui-Tuong Phongem v práci [15], stejně jako Phongův osvětlovací model 2.4. Metoda Phongoveho stínování je založena na znalosti normál ve vrcholu vykreslované plochy. Na rozdíl od Gouraudového stínování se ve nevypočítá barva, který se postupně interpoluje uvnitř plochy, ale normály se interpolují a osvětlovací model se počítá pro každý bod uvnitř této plochy.



Obrázek 2.7: Scéna s koulí vykreslená pomocí Phongova stínování[16].

Z obrázku 2.7 lze zpozorovat, že Phongovo stínování výborně zobrazuje zakřivené plochy. Používání Phongova stínování má ovšem úskalí v tom, že interpolace normál a výpočet osvětlovacího modelu v každém bodě je výpočetně náročné.



Obrázek 2.8: Srovnání konstantního, Gouradova a Phongova stínování[16].

Na obrázku 2.8 je porovnání Konstantního, Gouradova a Phongova stínování. Nejrychlejší z uvedených metod je metoda konstantního stínování, která se ovšem nepoužívá pro nedostatečně kvalitní výsledky. Druhá je metoda Gouradova, kterou implementují grafické akcelerátory, má nejlepší poměr fotorealismu a zobrazované rychlosti. Pro scény, které jsou zaměřené na kvalitu výsledného obrazu a nehledí se až tak na rychlost vykreslování, se ovšem používá stínování Phongovo, protože má nejlepší výsledky.

2.6 Zobrazovací metody

Tato kapitola se bude zabývat technikami zobrazení scény, které se snaží o určitou míru fotorealističnosti a při zobrazovaní scény zahrnout do výpočtu vzájemné působení (stín, odraz světla) objektů ve scéně.

Globální osvětlovací techniky se snaží, různými aproximacemi, najít řešení zobrazovací rovnice, která bude popsána v následující kapitole 2.6.

Globální zobrazovací techniky se dělí na dva typy zobrazovaní - pohledově nezávisle řešení, kdy se vypočítá osvětlení všech ploch ve scéně (například pomocí metody radiozity 2.6) nebo pohledově závisle řešení, kdy se vypočítá osvětlení pouze pro určitý směr - například metoda distribuovaného sledování paprsku 2.6.

Zobrazovací rovnice

Zobrazovací rovnici (anglicky rendering equation) poprvé formuloval James T. Kajiya roku 1986 v [13]. Řešením této rovnice je vycházející radiance v jakémkoliv bodě x a v libovolném směru ω. Zobrazovací rovnice v originálním znění pro výpočet radiance L_o z bodu x ve směru $\vec{\omega}$ se vypočítá jako:

$$L_o(x,\vec{\omega}) = L_e(x,\vec{\omega}) + \int_\Omega f(x,\vec{\omega_r},\vec{\omega_i}) L_o(x,-\vec{\omega_i}) \cos\Theta \,\mathrm{d}\vec{\omega_i} \tag{2.9}$$

kde:

- $L_e(x,\vec{\omega})$ je radiance vyzařovaná z bodu x (pokud je bod x světelným zdrojem)
- integrál je integrován přes polokouli Ω se středem v bodě x a představuje odraženou radianci v bodě x a směru ω
- $f(x,\vec{\omega},\vec{\omega_i})$ je dvousměrová odrazová distribuční funkce - BRDF
- $L_o(x,-\vec{\omega_i})$ je radiance která přichází do bodu x ze směru $-\vec{\omega_i}$
- $\cos\Theta$ je úhel mezi směrem $\vec{\omega_i}$ a normálou povrchu $\vec{n}$ v bodě x

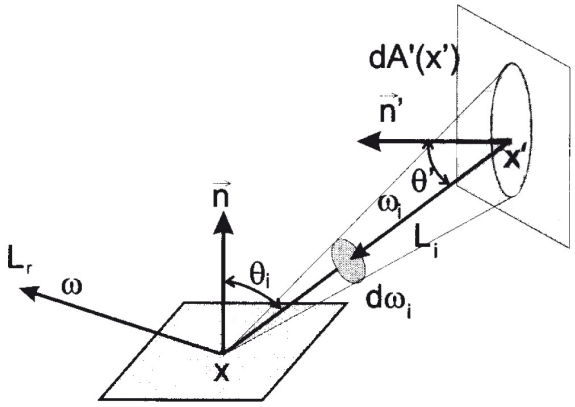
Řešení zobrazovací rovnice pro polokouli Ω může být nepraktická nebo obtížná. Proto je výhodnější formulace zobrazovací rovnice, která udává odraženou radianci jako integrál přes přispívající plochy [12].

Na obrázku 2.9 je zobrazena geometrie, pomoci níž se rovnice 2.9 aproximuje na tvar:

$$L_o(x,\vec{\omega}) = L_e(x,\vec{\omega}) + \int_S f(x,x'\to x,\vec{\omega}) L_i(x'\to x) V(x,x') G(x,x') \,\mathrm{d}A' \tag{2.10}$$

kde:

- integrál je zintegrovaný přes přispívající plochy S a představuje odraženou radianci v bodě x a směru ω
- $x' \to x$ je aproximace vektoru $\vec{\omega_i}$ z bodu x' do bodu x
- $V(x,x')$ určuje viditelnost (visibility term) bodu x z bodu x' a naopak, kdy pokud jsou si body navzájem viditelné, hodnota této funkce je 1, v druhém případě 0



Obrázek 2.9: Geometrie pro zobrazovací rovnici vyjádřenou pomocí plošek [12].

- $G(x, x')$ je tzv. geometricky člen (geometry term), který má v sobě zahrnuty koeficienty, které vyjadřují promítnutí radiance po vyzáření z bodu x' a po dopadu do bodu x. Geometrický člen má tvar:

$$G(x, x') = \frac{(\vec{\omega}.\vec{n'})(\vec{\omega'}.\vec{n})}{\|x' - x\|^2} \tag{2.11}$$

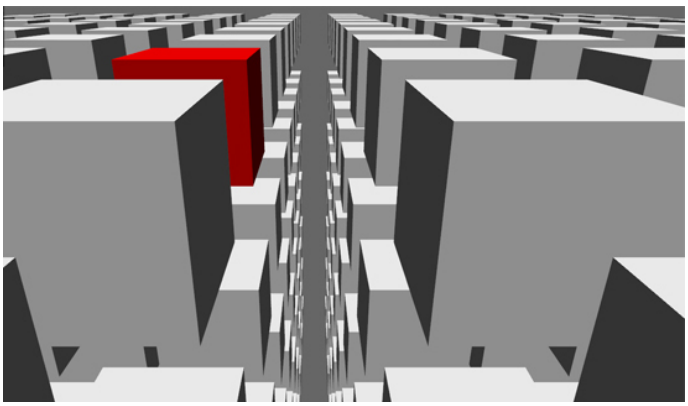
Rovnice 2.10 říká, že radiance vycházející z bodu x ve směru ω se vypočítá jako součet radiance vyzařující v bodě x s součtem přicházejících radiancí ze všech ostatních ploch, které jsou viditelné z bodu x vynásobených dvousměrovou distribuční funkcí BRDF. Problém ovšem nastane při získávání hodnoty vycházející radiance z bodu x' do bodu x, která je neznámá a musí se stejným způsobem vypočítat.

Metoda vrhání paprsku

Metoda vrhání paprsku (anglicky raycasting) je jedna ze základních zobrazovacích metod. Z této metody vychází další zobrazovací metody jako je například raytracomg nebo photong mapping. Tato metoda byla poprvé představena již v roce 1968 [1] Arturem Appelem.

Na rozdíl od reálného světa, kdy se světelné paprsky šíří ze světelných zdrojů pomocí různých odrazů až k pozorovateli, raycasting využívá opačného principu, kdy vysílá paprsek od pozorovatele přes průmětnu směrem do scény, hledá první objekt s kterým se protne a jeho barvu přiřadí pixelu, z kterého se vysílal paprsek.

Této metodě se také říká metoda sledování paprsku prvního řádu[12], protože zobrazuje pouze bod na povrchu nejbližšího tělesa zasaženého paprskem vyslaným směrem od pozorovatele. V daném bodě se určí barva pomocí libovolného zobrazovacího modelu - například Phongova. Tento algoritmus je rychlejší než metoda sledování paprsku, protože nepočítá s dalšími paprsky.



Obrázek 2.10: Scéna zobrazena pomocí raycastingu

Základní varianta raycastingu počítá s modelem popisující povrchy objektů ve scéně. Na obrázku[1] 2.10 lze vidět zobrazenou scénu vykreslenou raycastingem, popsanou velkým množstvím krychlí.

Metoda vrhání paprsku (s mírnou úpravou) se ve velké míře používá v medicíně při zobrazování volumetrických dat získaných například z tomografie, tato metoda se nazývá volume ray casting.

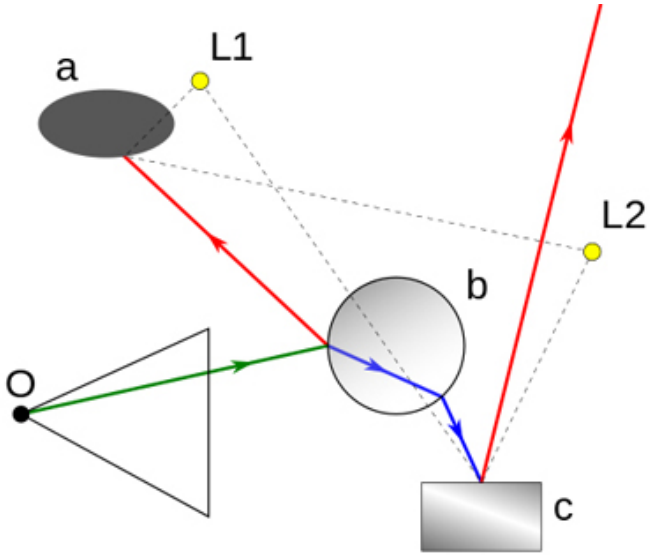
Na rozdíl od metody zobrazování scény rasterizací trojúhelníků, není u této metody potřeba řešit viditelnost, protože se vykreslí pouze objekt, který je v popředí. V základní variantě chybí v zobrazené scéně stíny. Touto metodou se zobrazované scéně nedocílí žádných fotorealistických efektů.

[1] zdroj: http://blogs.aerys.in/jeanmarc-leroux/2012/05/30/new-minko-2-feature-ray-casting/

Metoda sledování paprsku

Metoda sledování paprsku (anglicky Raytracing) je jedna z fotorealistických zobrazovacích technik. Metodu poprvé uvedl Whitted v roce 1980 [18]. Jedná se o rekurzivní metodu a tato metoda je na rozdíl od raycastingu schopna zobrazit zrcadlové odrazy jiných těles a zpracovat vržené stíny.

Metoda sledování paprsku vychází podobně jako raycasting ze stejného principu, kdy se světelné paprsky vysílají přes průmětnu směrem do scény a hledají průsečík s prvním tělesem ve scéně. Narozdíl od raycastingu se této metodě říká metoda sledování paprsku vyššího řádu [12], protože nekončí pouhým průsečíkem s prvním tělesem, ale v průsečíku vytvoří další paprsky, které rekurzivně znova vyhodnocuje.



Obrázek 2.11: Základní schéma raytracingu[26].

Základní schéma principu raytracingu je naznačeno na obrázku 2.11. Na tomto obrázku lze vidět, že paprsek vyslaný od pozorovatele O se dále dělí na další paprsky. Raytracing využívá více druhů paprsku a těmi druhy paprsku jsou:

- *Primární paprsek* (*primary ray*), je vysílán z místa pozorovatele směrem do scény. Počet primárních paprsku je roven počtu pixelů v zobrazovací rovině. (V předchozím obrázku se jedná o zelený paprsek)
- *Sekundární paprsek* (*secondary ray*), je vytvořen po dopadnutí primárního nebo sekundárního paprsku na objekt ve scéně. V bodě dopadnutí primárního paprsku se vytvoří *odražený* (*reflected*) paprsek (červený paprsek v obrázku) a *lomený* (*refracted*) paprsek (modrý paprsek v obrázku). Počet sekundárních paprsku závisí na hloubce rekurze a jejich počet je mnohem větší než primárních paprsků, protože při každém dopadu primárního (sekundárního) se vytvoří další dva sekundární paprsky.

- *Stínový paprsek* (*shadow ray*) se vysílá z každého bodu dopadu primárního nebo sekundárního paprsku na těleso směrem k světelnému zdroji. Má za úkol určit, zda-li na daný bod světelný zdroj dosvítí, nebo mu v cestě stojí nějaká překážka. Výpočet stínového paprsku je jednodušší v tom, že není třeba určit pořadí clonících objektů, pouze zda-li má paprsek volný prostor mezi bodem a světelným zdrojem. Pokud je světelný zdroj nezastíněn žádným tělesem, je tento světelný zdroj zahrnut do vyhodnocení osvětlovacího modelu v daném bodě. Z každého bodu se vysílá tolik paprsků, kolik je světelných zdrojů. Na předchozím obrázku je stínový paprsek označen čárkovanou čárou.

Algoritmus 2.6.1. Sledování paprsku vyššího řádu [12]. *Mějme funkci SledujPaprsek (paprsek R, hloubka rekurze H):*

1. Nalezni průsečík P paprsku R s nejbližším tělesem ve scéně.
2. Pokud průsečík P neexistuje (paprsek opustil prostor scény), přiřaď paprsku R barvu pozadí a skonči.
3. Ke každému světelnému zdroji vyšli z bodu P stínový paprsek a pokud k němu paprsek dorazí, označ světelný zdroj jako nezakrytý,
4. Vyhodnoť příspěvky osvětlení v bodě P od všech nezakrytých světelných zdrojů,
5. Pokud hloubka H nepřekročila maximální hloubku sledování vyšli
   - odražený paprek R_R voláním SledujPaprsek (R_R, $H + 1$)
   - lomený paprsek R_T voláním SledujPaprsek (R_T, $H + 1$)
6. Paprsku R přiřaď výslednou barvu jako součet příspěvků osvětlení, barvy odraženého paprsku R_R a barvy lomeného paprsku R_T,

Z výše uvedeného algoritmu je patrné, že časová náročnost je úměrná hloubce rekuze H. Při hloubce rekurze $H = 1$, bychom získali vylepšenou metodu vrhání paprsku, popsanou v kapitole 2.6 , o stínové paprsky. Díky výpočetní nezávislosti každého vyslaného paprsku, je možné tento algoritmus provádět paralelně, čímž získáme určité zrychlení závislé na počtu paralelních výpočtů.

Páty bod u výše zmíněného algoritmu se dá upravit tak, že se odražený a lomený paprsek vyšle pouze, když příspěvek světla nových paprsků je vetší než předem určená minimální hodnota.

Pro určení výsledné barvy paprsku se používá libovolný osvětlovací model například Phongův.

Phongův osvětlovací model (2.4) je nutné rozšířit o intenzitu odraženého a lomeného paprsku. Výsledný součet výpočtu rozšířeného Phongova osvětlovacího modelu má tvar:

$$I = I_s + I_d + I_a + I_r + I_t \tag{2.12}$$

V tomto tvaru jsou obsažené známé složky pro zrcadlové I_s, difúzní I_d a okolní (ambientní) I_a osvětlení. Přidány jsou složky I_r pro intenzitu odražené paprsku a I_t pro intenzitu lomeného paprsku, obě dvě složky se vypočítají jako:

$$I_r = r_s I_R \tag{2.13}$$

Kde I_R je výsledná intenzita paprsku R_R, r_s je koeficient zrcadlového odrazu daného tělesa. Výpočet intenzity I_t paprsku R_T je podobný, může být však doplněn o další koeficient charakterizující útlum závisející na uražené vzdálenosti paprsku uvnitř tělesa [12].

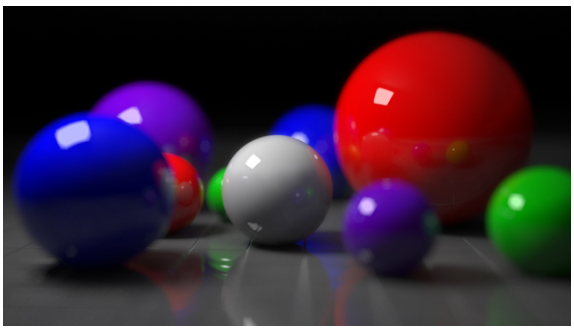
Distribuovaný raytracing

Pomocí raytracingu se dají vykreslit efekty, které se u jiných (základních renderovacích) metod dopočítávají velmi těžko jako stíny, odražené nebo lomené paprsky. Jeden velký problém však metoda raytracingu má a tím problémem jsou ostré stíny u plošných světelných zdrojů.

Ostré stíny se vyznačují tvrdým (náhlým) přechodem mezi osvětlenou oblastí a oblastí, která leží ve stínu. Metoda raytracingu vykresluje scény s ostrými stíny, protože stínové paprsky, které určují zda-li zobrazovaný bod leží ve stínu či nikoliv, jsou počítány přesně (bod buď leží nebo neleží v osvětlené oblasti) a hodnoty nepřímého osvětlení nejsou počítány přesně, ale používá se ambientní složka v phongově osvětlovacím modelu.

Problém ostrých stínů řeší upravená metoda raytracingu z názvem *distribuovaný raytracing*. Tuto metodu poprvé popsal Robert L. Cook roku 1984 v [4]. Tato metoda spočívá v generování ne jednoho přesného paprsku, ale svazku N paprsků, kolem původního přesného paprsku. Tyto paprsky mají náhodné směry dané distribuční funkcí specifickou pro každý povrch (viz 2.4). Tímto způsobem se pomocí stochastického vzorkování (Monte Carlo) aproximoval lokální osvětlovací model 2.2.

Distribuovaný raytracing přispívá k fotorealistickému vzhledu výsledného obrazu těmito vlastnostmi [17]: rozmazáním odražených a lomených objektů, měkkými stíny, hloubkou ostrosti (depth of field) a rozmazání pohybem, kdy stochastické vzorkování paprsku se děje nejen v prostoru, ale i v čase.



Obrázek 2.12: Scéna vykreslena pomocí distribuovaného Raytracingu[25].

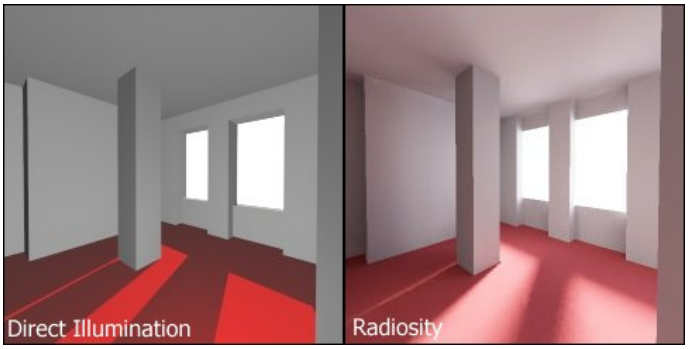
Obrázek 2.12 znázorňuje scénu vykreslenou rendererem V-Ray, který je součástí modelovacího softwaru Rhinoceros 3D. Problém distribuovaného raytracingu je jeho enormní časová náročnost, kdy je pro jeden paprsek potřeba vypočítat dalších N paprsků.

U metody sledování paprsků není třeba, na rozdíl od metody zobrazování scény rasterizací trojúhelníku, řešit viditelnost, protože stejně jako u raycastingu, je zobrazeno pouze první těleso ve scéně, které paprsek protíná. Díky stínovým paprskům, jsou ve scéně korektně zobrazeny stíny, takže není nutné je složitými způsoby dopočítávat.

Metoda sledování paprsků obsahuje fotorealistické efekty jako jsou vržené stíny, odražené a lomené objekty. Problém této metody je ten, že nepřímé osvětlení se nevypočítává přesně, ale je určeno globálně pro celou scénu, čímž se do scény dostává určitá nepřesnost. Dalším problémem jsou ostré stíny, které se snaží řešit vylepšená metoda o distribuování paprsků, čímž se určitou mírou aproximuje zobrazovací rovnice a docílí se lepších výsledků.

Radiozita

Metoda sledování paprsku, popsána v kapitole 2.6, kvalitně zobrazuje zrcadlové odrazy a průhledné objekty, výsledné obrazy mají však k fyzikálně věrné simulaci daleko. Tato metoda (v základní podobě) počítá pouze s bodovými zdroji světla a jako nepřímé difusní osvětlení používá neměnný ambientní člen. Pro dosažení co největší věrnosti fotorealistických snímku s realitou, je nutné použít metodu, která korektně simuluje šíření světla scénou [12]. Rozdíl mezi scénou, která je zobrazena pomocí přímého osvětlení - tvoří tvrdé stíny a měkkými stíny, které vytváří metoda radiozity, jsou vidět na obrázku 2.13.



Obrázek 2.13: Rozdíl mezi přímým osvětlením a radiozitou[24].

První pokus o vytvoření metody pro korektní šíření světla scénou je inspirována poznatky z oblasti výpočtů tepelného záření a byl popsán v roce 1984 vědci Goralová, Torrance, Greenberg a dalšími [7]. Radiozita vychází ze zákona o zachování energie, předpokládá přenos světelného záření v uzavřené scéně, prostředí ve scéně netlumí procházející světlo a všechny objekty jsou zcela neprůhledné [12].

Radiozitní rovnice je popsána vztahem:

$$B(x) = E(x) + \rho(x) \int_S B(x')G(x, x')\,\mathrm{d}x' \tag{2.14}$$

kde:

- $B(x)$ je radiozita vyzařovaná povrchem v bodě x
- $E(x)$ vyzařovaná energie povrchu v bodě x
- $\rho(x)$ je difúzní odrazivost plochy v bodě x. Hodnota udává množství energie odražené, při dopadu energie na bod x. Nabývá hodnot 0 (kompletní pohlcení) až 1 (kompletní odrazivost - plocha nepohltí žádnou energii).
- $G(x, x')$ je geometrický člen, který obsahuje informace, které se týkají vždy dvojice ploch x a x'. Obsahuje informace jako vzájemnou viditelnost, vzdálenost, natočení normál.

- S je plocha scény.

Pro netriviální scény je analytické řešení radiozitní rovnice, v základním znění - 2.14, tak složité, že je až nemožné. Proto bylo nutné tuto rovnici zjednodušit a povrch scény aproximovat na množinu rovinných plošek, kdy hodnota radiozity $B(x)$ je na celém povrchu této plošky konstantní. Přeformulovaná rovnice radiozity na její zjednodušenou diskrétni formu vypadá následovně:

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij} \tag{2.15}$$

kde:

- B_i je radiozita plošky i
- E_i je radiozita vyzařovaná ploškou i
- suma reprezentuje součet radiozit všech plošek ve scéně, které přichází na plošku i
- F_{ij} je konfigurační faktor (form factor) mezi ploškou i a ploškou j. V diskrétní formě radiozitní rovnice nahrazuje geometrický člen $G(x, x')$ a udává kolik energie vyzářené ploškou j je přijato ploškou i.

K řešení a zobrazení scény pomocí diskrétní radiozitní rovnice 2.15, je zapotřebí model, který reprezentuje scénu soustavami ploch doplněných o difúzní odrazivost ρ_i plochy a zářivost E_i dané plochy.

Prvním krokem je rozdělení povrchu scény na síť rovinných plošek. Jemnost dělení scény na malé plošky určuje kvalitu výsledného zobrazení. Při hrubých plochách bude kvalita značně horší, než při velmi jemném dělení. Naopak velmi jemné dělení scény zapříčiňuje velkou časovou a paměťovou složitost, je tedy nutné najít určitý kompromis mezi věrohodností výsledného obrazu a časovou náročností.

Dalším krokem je určení konfiguračních faktoru plošek. Velikost konfiguračního faktoru F_{ij} mezi ploškou i a j je ovlivněna několika faktory:

- Velikostí plošek - čím bude ploška j větší, tím víc bude ovlivňovat plošku i a tím bude konfigurační faktor F_{ij} větší.
- Vzájemnou polohou plošek - pokud bude normálový vektor plošky j kolmý na normálový vektor plošky i bude konfigurační faktor menší, než když budou normálové vektory obou plošek rovnoběžné.
- Vzájemnou polohou plošek - pokud bude mezi ploškou i a ploškou j nějaká ploška (budou v zákrytu), bude konfigurační faktor menší, než když mezi oběma ploškami nebude žádná ploška.

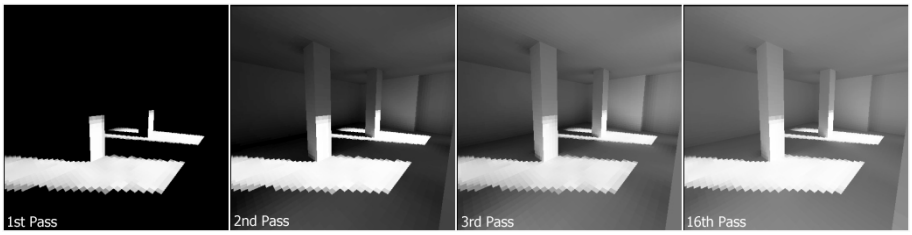
Po rozdělení povrchu scény a vypočtení konfiguračních faktoru, je již možné vypočítat hodnoty radiozity B_i dané plošky, jako soustavu n rovnic pro n neznámych B_1 až B_n [12].

Existuje několik možností počítání radiozity B_i ze soustavy rovnic, buď se využijí postupy známe z numerické matematiky jako Gauss-Seidelova nebo Jakobiho iterační metoda anebo se využije speciální řešení jako například progresivní radiozita, hierarchická radiozita a podobně [12].

Využití numerických metod, Gauss-Seidelovy nebo Jakobiho iterační metody, je v počítačové grafice nevhodné, protože není možné ovlivnit, kdy se do výpočtu zahrnou silné světelné zdroje. Tyto metody pouze dokáží zajistit, že světelné zdroje budou do výpočtu zahrnuty, ale není možné určit kdy [12].

Progersivní radiozitu představili Cohen, Wallace a Grennberg [3]. Řeší problém numerického výpočtu a umožňuje zobrazení výsledku ještě před jeho kompletním dopočítáním.

Progresivní radiozita postupně "vystřeluje"radiozitu z plošky, která má nejvíce nevyzářené energie. V jednom iteračním kroku metoda nalezne plochu j s největší nevyzářenou energií, spočítá konfigurační faktor F_{ij} pro všechny zbývající plošky (u této metody není nutné počítat všechny konfigurační faktory před zahájením výpočtu), "vystřelí"radiozitu z plochy j směrem do scény a plošky, které jsou touto radiozitou zasaženy, se v dalším iteračním kroku stanou zdroji světla.



Obrázek 2.14: Dílčí výsledky progresivní radiozity[24].

Na obrázku 2.14 lze vidět, jak progresivní radiozita postupně postupuje od světelných zdrojů směrem do scény a poté se odráží od osvícených ploch.

Obrázek 2.15: Scéna vykreslena pomocí radiozity[3].

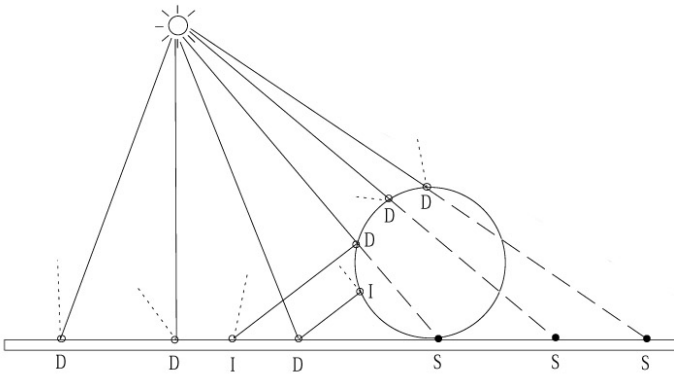
Po vypočtení radiozity pro každou plochu, je výsledek zobrazen například pomocí raycastingu. Pomocí radiozity je možné docílit dobrých fotorealistických výsledků obrazu. Vyniká hlavně ve stínech, metoda není závislá (oproti raytracingu) na úhlu pohledu. Metoda je ovšem poměrně časově náročná, má nutnost rozdělit scénu na malé plošky a nepočítá s difusní složkou světla [12].

Photon mapping

Photon mapping je dvouprůchodová metoda globálního osvětlování, kterou představil Henrik Wann Jensen v roku 1996 [10]. Tato metoda se snaží o řešení zobrazovací rovnice vhodnou aproximací. Tato metoda, na rozdíl od radiosity, nepotřebuje rozdělení scény na ploškový tvar, počítá se scénou tak jak je. Scény vykreslené pomocí Photon mappingu realisticky simulují interakci světla s různými objekty, věrně zobrazují lom světla skrz průhledné objekty jako je voda nebo sklo, čímž vytvářejí věrohodné kaustiky.

První průchod - Tvorba fotonové mapy

V prvním průchodu se ze světelných zdrojů vystřelují fotony, které odpovídají zlomku energie daného světelného zdroje. U každého fotonu je, podobně jako u metody sledování paprsku, prošetřována cesta kudy foton putuje po scéně. Oproti metodě sledování paprsku se vždy při nárazu do nějaké plochy uloží hodnota úbytku světelného toku $\Delta\Phi_p(x, \vec{\omega_p})$ a směr odraženého/lomeného fotonu do fotonové mapy. Dál se foton odrazí nebo lomí skrze objekt. Směr odrazu (lomu) a intenzita fotonu je určena pomoci dvousměrné odrazové distribuční funkce specifické pro každý povrch.



Obrázek 2.16: První průchod - přímy paprsek - D, nepřímy - I, stínový - S [10]

Fotonová mapa je abstraktní datová struktura, v které jsou uložené fotony na povrchu těles. Při první fázi je tato struktura pouze pole, do kterého se postupně zapisují data jako: xyz souřadnice, energie fotonu, směr fotonu a typ. Typ fotonu může být buď přímý, nepřímý (je tvořen odrazem) nebo stínový. Rozdělení na přímý reps. nepřímý foton je využito při druhé fázi, kdy se pomocí těchto fotonů určuje přímé resp. nepřímé osvětlení. Stínové fotony jsou používány pro snížení počtu stínových paprsku a tím pro urychlení zobrazovaní. Více o použití stínových fotonů je v [11].

Po ukončení prvního průchodu a vypočítání někdy i milionů fotonů, je nutné mít datovou strukturu v které se bude rychle vyhledávat, pro tuto metodu je doporučený kd-strom, který se poté z daného pole vytvoří. Pro photon mapping se většinou používají dvě fotonové mapy: jedna globální, pro odražené fotony, a druhá pouze pro fotony, které vychází skrze objekty a vytvářejí kaustiky. Rozdělením na dvě fotonové mapy, se získá určité zrychlení, kdy při výpočtu kaustik se vyhledává pouze v mapě pro kaustiky.

Druhý průchod - renderování

Druhý průchod photon mappingu je podobný klasické metodě sledování paprsku, kdy jsou paprsky vysílány přes průmětnu směrem do scény a postupně vyšetřovány v rámci scény. Pokud vyšetřovaný paprsek narazí na difúzní povrch, je hodnota radiance v daném bodě vypočítána pomocí fotonové mapy, pokud nenarazí je paprsek vyhodnocován dál do scény (pomocí odrazu a lomu) dokud na difúzní povrch nenarazí.

Metoda photon mappingu aproximuje řešení zobrazovací rovnice pomocí uložených úbytků světelného jasu fotonů, získaných v první fázi. Radiance $L_o(x, \vec{\omega})$, která vychází v bodě průsečíku x ve směru k pozorovateli $\vec{\omega}$, se vypočítá pomocí zobrazovací rovnice (popsané v kapitole 2.6) jako součet radiance vyzařované $L_e(x, \vec{\omega})$ a radiance odražené $L_r(x, \vec{\omega})$, která se vypočítá jako:

$$L_r(x, \vec{\omega}) = \int_\Omega f(x, \vec{\omega}, \vec{\omega_i}) L_i(x, \vec{\omega_i}) \cos\Theta \, \mathrm{d}\vec{\omega_i} \tag{2.16}$$

Pokud se nahradí přicházející radiance $L_i(x, \vec{\omega_i})$ vztahem:

$$L_i(x, \vec{\omega_i}) = \frac{\mathrm{d}^2\Phi_i(x, \vec{\omega_i})}{\cos\Theta \, \mathrm{d}\vec{\omega_i} \, \mathrm{d}A} \tag{2.17}$$

který vyjadřuje vztah mezi radiancí a světelným tokem. Když se tento vztah dosadí do rovnice 2.16 a patřičně vykrátí, získá se radiance odražená jako:

$$L_r(x, \vec{\omega}) = \int_\Omega f(x, \vec{\omega}, \vec{\omega_i}) \frac{\mathrm{d}^2\Phi_i(x, \vec{\omega_i})}{\mathrm{d}A} \tag{2.18}$$

tento integrál je možné odhadnout jako sumu:

$$L_r(x, \vec{\omega}) = \int_\Omega f(x, \vec{\omega}, \vec{\omega_i}) \frac{\mathrm{d}^2\Phi_i(x, \vec{\omega_i})}{\mathrm{d}A} \approx \sum_{p=1}^{N} f(x, \vec{\omega}, \vec{\omega_p}) \frac{\Delta\Phi_p(x, \vec{\omega_p})}{\Delta A} \tag{2.19}$$

Pokud se v bodě průsečíku x hledá N nejbližších fotonů, budou tyto fotony ležet uprostřed koule se středem v bodě x, poté je možné aproximovat $\Delta A = \pi r^2$, čímž se získá konečný tvar aproximované zobrazovací rovnice pro získání odražené radiance $L_r(x, \vec{\omega})$ v bode x a ve směru ω:

$$L_r(x, \vec{\omega}) \approx \frac{1}{\pi r^2} \sum_{p=1}^{N} f(x, \vec{\omega}, \vec{\omega_p}) \Delta\Phi_p(x, \vec{\omega_p}) \tag{2.20}$$

Pro každý paprsek se tedy počítá průsečík s tělesem a poté se vyhledá N nejbližších fotonu, z kterých se vypočítá barva daného paprsku. Vyhledávání N nejbližších fotonů pomocí hledání v kouli, může selhávat v rozích scény, kdy se mohou zahrnout fotony z jiné roviny. V těchto případech se používá jiný tvar než koule, jako například válec, kvádr, disk.

Pokud se photon mapping počítá vícekrát pro stejnou scénu je výsledek první fáze photon mappingu vždy stejný - fotonová mapa se pro scénu počítá pouze jednou a může být uložena pro příští použití. Pokud se při druhé fázi změní pohled, je nutné znovu celou druhou fázi renderovat znovu.

Podle výsledku uvedených v [10] photon mapping dosahuje stejné kvality výsledného obrazu v řádově menších časech jako radiozita.

2.7 Průsečík paprsku s objekty scény

V počítačové grafice je často vyhodnocován průsečík paprsku s objekty scény jako třeba s krychlí, koulí, rovinou, nebo trojúhelníkem.

Protože model objektů scény bývá popsán jako síť trojúhelníku, je jeden z nejčastějších a časově nejkritičtějších výpočtů průsečíku výpočet průsečíku paprsku s trojúhelníkem.

Pokud máme paprsek vycházející z počátku (*origin*) O ve směru (*direction*) $\vec{D}$ a trojúhelník s vrcholy A, B, C je možné vypočítat průsečík paprsku s trojúhelníkem jako systém rovnic určený vztahem:

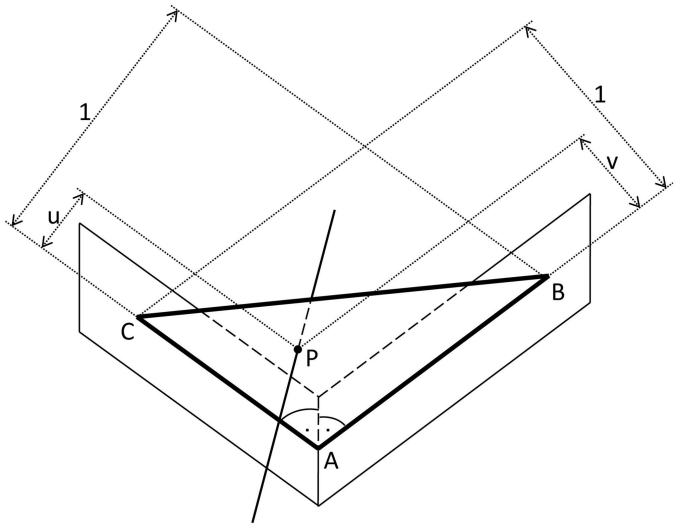
$$O + t.\vec{D} = A + u.(B - A) + v.(C - A) \tag{2.21}$$

kde t je parametr určující vzdálenost paprsku od od počátku O ve směru $\vec{D}$ a parametry u a v jsou barycentrické souřadnice. Pro určení, zda-li průsečík paprsku s trojúhelníkem leží uvnitř trojúhelníku, musí platit tyto pravidla:

$$\begin{aligned} 0 &\leq t \leq t_{max} \\ u &\geq 0 \\ v &\geq 0 \\ (u + v) &\leq 1 \end{aligned} \tag{2.22}$$

Metody pro nalezení průsečíku paprsku s trojúhelníkem se odlišují na dva základní typy podle toho jak naloží se systémem rovnic uvedeným v 2.21, první řeší systém rovnic přímo pomocí Cramerova pravidla a druhé řeší systém rovnic pomocí vhodné substituce.

Jedna z nejrychlejších metod patří do metod řešených substitucí a byla představena panem J. Havlem v [9].



Obrázek 2.17: Kolmé roviny k trojúhelníku použité pro zjištění barycentrických souřadnic [9]

Tato metoda spočívá v určení barycentrických souřadnic průsečíku, s rovinou trojúhelníku, pomocí dvou předpočítaných rovin, které jsou kolmé na rovinu trojúhelníku. Trojúhelník je tedy určený trojicí bodů A, B, C, rovinou trojúhelníku $(\vec{N}, d)$ a dalšími dvěmi předpočítanými rovinami $(\vec{N_1}, d_1)$ a $(\vec{N_2}, d_2)$, které se vypočítají podle:

$$\begin{aligned} \vec{N_1} &= \frac{\vec{AC} \times \vec{N}}{\vec{N}^2}, \quad d_1 = -\vec{N_1}.A \\ \vec{N_2} &= \frac{\vec{N} \times \vec{AB}}{\vec{N}^2}, \quad d_2 = -\vec{N_2}.A \end{aligned} \tag{2.23}$$

Pro zjištění průsečíku paprsku s rovinou trojúhelníku jsou použité výpočty:

$$\begin{aligned} det &= \vec{D}.\vec{N} \\ t' &= d - (O.\vec{N}) \\ P' &= det.O + t'.\vec{D} \\ u' &= P'.\vec{N_1} + det.d1 \\ v' &= P'.\vec{N_2} + det.d2 \end{aligned} \tag{2.24}$$

pro určení, zda-li průsečík nastal uvnitř trojúhelníku, musí platit podmínky:

$$\begin{aligned} sign(t') &= sign(det.t_{max} - t') \\ sign(u') &= sign(det - u') \\ sign(v') &= sign(det - u' - v') \end{aligned} \tag{2.25}$$

pokud tyto podmínky platí, jsou vypočítány barycentrické souřadnice u, v a parametr t podle:

$$\begin{aligned} t &= \frac{1}{det}.t' \\ u &= \frac{1}{det}.u' \\ v &= \frac{1}{det}.v' \end{aligned} \tag{2.26}$$

Barycentrické souřadnice u a v jsou určené k interpolaci normál, nebo uv texturovacích souřadnic případně i bodu průsečíku. Pro zjištění průsečíku je však vhodnější dosadit parametr t do rovnice přímky v prostoru:

$$P = O + t.\vec{D} \tag{2.27}$$

čímž se získá bod v prostoru, který označuje průsečík vrhaného paprsku s trojúhelníkem.

2.8 Software pro fotorealistické zobrazování

Tato podkapitola bude popisovat některé softwarové prostředky pro tvorbu fotorealistického obrazu. Většinou jde o programy, které neslouží pouze čistě k renderování fotorealistického obrazu, ale i k modelování scény, vytváření animací a dalších věcí.

POV-ray

POV-ray (z anglického Persistence of Vision Raytracer) je multiplatformní renderovací software, který vykresluje scénu pomocí metody sledování paprsku.

Pro popis scény v programu POV-ray je použit speciální jazyk pro popis scény SDL - scene description language. Popis komplexnějších scén může být tímto způsobem zdlouhavý, proto existují plug-iny do jiných modelovacích programů, které popis scény generují - například PovAnim pro Blender.

Vykreslení scény probíhá pomocí metody sledování paprsku. Novější verze vykreslují scénu pomocí metody radiozity nebo photon mappingu [23].

Blender

Blender je volně dostupný opensource modelovací nástroj, který slouží k vytváření vizuálních efektů, animovaných filmů nebo videoher [20].

Pro specifikaci scény slouží v Blenderu klasický modelovací nástroj. Dají se v něm modelovat postavy, animace i klasické modely.

Scéna se vykresluje pomocí metody sledování paprsku. Blender umožňuje spustit metodu sledování paprsku pomocí technologie CUDA paralelně na grafické kartě dokonce i na více grafických kartách [6].

Autodesk 3DS max

Autodesk 3ds Max, kdysi známe jako 3D studio Max, je modelovací nástroj pro tvorbu animací, modelů a vyrenderovaných obrázků.

Scéna se v tomto programu jako v ostatních modelovacích nástrojích. Scéna je vykreslena pomocí kvalitní metody sledování paprsku[19].

2.9 Další použité techniky

Protože na popsání dalších principů a metod, použitých v této práci, již není místo, jsou zbylé metody popsány už jen krátkým popisem s odkazem na literaturu.

Antialiasing

Ve scénách zobrazovaných v počítačové grafice se vyskytují nechtěné útvary – aliasy. Co to alias je a jak se potlačuje je popsáno v [12].

Textury

Pro popis barvy objektu scény je možné využít textury, kdy objekt nemá pouze jednu barvu, ale pro každý bod na objektu je specifikována barva v textuře. Co to textury jsou a způsob výpočtu barvy z textury pomocí tzv. uv souřadnic je v [12].

Rozdělení do podprostoru

Průsečík paprsku s mnoha objekty scény může být zdlouhavý. Pro urychlení se dá prostor rozdělit na menší části a průsečík paprsku počítat pouze s objekty uvnitř podprostoru. Existují tři typy rozdělení do podprostoru: BSP strom, oktalový strom a kd tree. Více info v [12].

Kapitola 3

Zhodnocení stavu a plán práce

3.1 Zhodnocení stavu

V současné době existuje v počítačové grafice velké množství zobrazovacích metod, které se mezi sebou hodně liší. Liší se například ve věrohodnosti výsledného obrazu nebo rychlosti vykreslení scény.

Nejrozšířenější zobrazovací metoda je metoda zobrazování scény rasterizací trojúhelníku, tato metoda je hodně rozšířená hlavně kvůli její poměrně nízké časové náročnosti oproti ostatním metodám, kdy je možné zobrazovat scénu v reálném čase.

Metody sledování paprsku mají oproti metodě rasterizace trojúhelníku realističtější výstup za cenu větší časové náročnosti. Tato metoda má ale svoje problémy, jako ostré stíny nebo nekorektně vypočítané nepřímé osvětlení.

Korektní šíření světla scénou řeší metody radiozity a photon mappingu. Tyto metody nejprve vypočítají osvětlení ve scéně a až poté zobrazují výsledek. Metoda photon mappingu má nejrealističtější výsledek zobrazení ze zde uvedených metod za cenu velmi velké časové náročnosti.

V současné době se používají fotorealistické metody pouze pro zobrazení animací, filmů a v oblastech, kde není nutnost zobrazit scénu v reálném čase. Problém který brání k většímu nasazení fotorealistických metod je ten, že jejich výpočet je velmi náročný. V dnešní době se sice objevují studie, kdy je v reálném čase zobrazována například metoda sledování paprsku. Jedná se ovšem o velmi malou a jednoduchou scénu nebo zobrazování je počítáno na velmi drahých grafických kartách.

3.2 Důvod tvorby práce

Bakalářskou práci z oblasti počítačové grafiky jsem si vybral, protože k počítačové grafice jsem měl vždy blízký vztah a chtěl bych se jí zabývat i v budoucnu. Na přednáškách předmětu základy počítačové grafiky mě zaujalo téma o fotorealistickém zobrazení a proto bych si chtěl vytvořit fotorealistický renderer.

V prvním plánu bylo vytvořit renderer, který zobrazuje scénu pomocí metody photon mappingu. Problém ovšem nastává v tom, že nemám dostatečné znalosti a zkušenosti na to, abych tento renderer vytvořil, protože metoda photon mappingu je poměrně složitá na implementaci. Proto jsem se rozhodl vytvořit renderer, který zobrazuje scénu pomocí metody sledování paprsku, na této metodě získat potřebné znalosti a zkušenosti a photon mapping si vyzkoušet případně v diplomové práci.

I když existuje mnoho softwaru ke kterým by se dal renderer přidat jako zásuvný modul, rozhodl jsem se vytvořit renderer úplně od základů, kdy jediné co si převezmu od existujícího softwaru je vyexportovaný soubor, který specifikuje zobrazovanou scénu. Právě na vytvoření rendereru kompletně od základu je snaha o zisk co nejvíce zkušeností z počítačové grafiky, kdy je nutné nastudovat a naimplementovat vše co souvisí se zobrazením scény.

3.3 Plán práce

V implementační části bakalářské práce by se měl vytvořit renderer, který bude mít tyto vlastnosti a parametry:

- metoda sledování paprsku
- vlastní vnitřní reprezentace scény
- rychlý výpočet průsečíku paprsku s trojúhelníkem
- zmenšení počtu výpočtů průsečíku paprsku s trojúhelníkem pomocí oktalového stromu
- textury
- řešení aliasu ve výsledném obraze

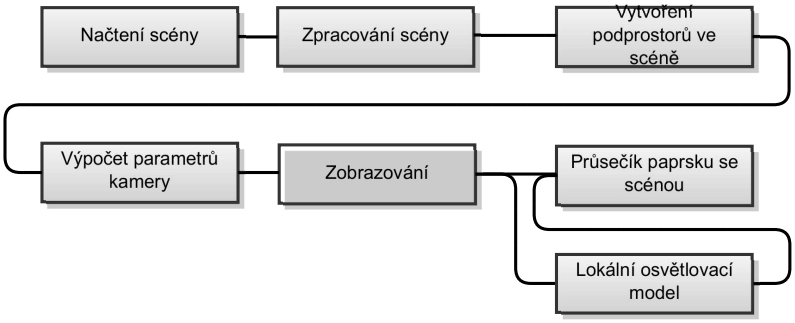
Kapitola 4

Tvorba Raytraceru

Tato kapitola se bude zabývat praktickou částí bakalářské práce. Bude v ní postupně vysvětlen postup vývoje raytraceru a budou probrány netriviální záležitosti z implementace.

4.1 Návrh programu

Architektura programu je rozdělena na funkční částí - bloky. Kooperaci bloků znázorňuje blokový diagram - obrázek 4.1.



Obrázek 4.1: Blokový diagram

Blok načtení scény načítá scénu ze souboru a ukládá ji do vnitřní reprezentace scény. Zpracováním scény se myslí vytvoření jedné trojúhelníkové sítě z modelu. Blok vytvoření podprostorů rozdělí scénu na podprostory pomocí oktalového stromu.

V další části se vypočítají parametry kamery - zobrazovací průmětny, pomocí které se vysílají paprsky do scény - zobrazování. Po vyslání paprsku do scény je nutné určit průsečík paprsku s objekty scény a poté v průsečíku vyhodnotit lokální osvětlovací model.

4.2 Načítání scény

Pro načítání scény ze souboru je použit souborový formát COLLADA. COLLADA (*collaborative design activity*) je formát souboru pro ukládání 3D objektů a animací. Byl vyvinut firmou Sony Computer Entertainment, od verze 1.4 (která je použita při tvorbě RayTraceru) je formát COLLADA spravován konsorciem Khronos Group Inc.

Formát COLLADA byl původně vyvinut jako formát pro přenášení dat mezi modelovacími nástroji, pro uložení je použito xml schéma. Výstup nebo načtení tohoto formátu podporují modelovací nástroje jako: 3ds max, blender, Maya. Byl zahrnut i v herních enginech jako: Unreal Engine, CryEngine nebo dalších programech jako například Google earth.

Formát souboru COLLADA specifikuje scénu složenou z modelů, které jsou reprezentovány jako trojúhelníková síť, k nim jsou přidané normály a uv texturovací souřadnice. Dále je v souboru možné uložit nebo načíst různé druhy světel, kameru, materiály, animace, fyziku a další.

Pro zjednodušení manipulace se souborem scény uložené podle formátu COLLADA, je v programu RayTracer použito aplikační rozhraní s názvem COLLADA DOM. Toto API umožňuje snadné načítání scény ze souboru a jeho použitím odpadla nutnost psát si vlastní analyzátor pro načítání objektů scény.

4.3 Vnitřní reprezentace scény

Scéna načtená ze souboru je postupně ukládaná do struktur.

Modely objektů jsou uloženy do pole struktur *structSceneObject*. U modelu objektů jsou uloženy vrcholy trojúhelníku specifikující model, normály, texturovací uv souřadnice, transformační matice a data specifikující materiál objektu - barvy nebo textury.

Světla jsou uloženy ve struktuře *light*, pozice a parametry jsou uloženy ve struktuře *cameraObject* a směr pohledu kamery je uložen ve struktuře *lookAtObject*.

Po načtení modelů, světel a kamery, jsou trojúhelníky všech objektů uloženy do pole struktur *triangleObject*, kde se uloží: vrcholy trojúhelníku, normály, uv souřadnice, objekt ke kterému patří a předpočítaná data pro vyhodnocení průsečíku s trojúhelníkem (viz 4.5). Pozice vrcholů trojúhelníků jsou transformována pomocí transformačních údajů uložených spolu s modelem objektu.

4.4 Dělení prostoru - oktalový strom

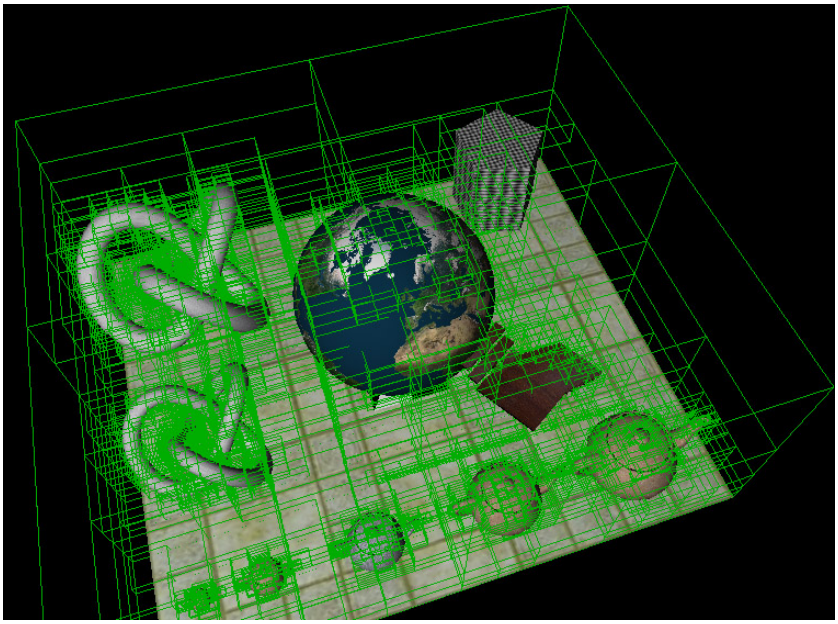
Pro urychlení výpočtů průsečíku paprsku s trojúhelníky, je prostor scény rozdělen na podprostory pomocí metody *oktalového stromu*.

Pro tvorbu oktalového stromu, je nejprve určena ohraničující oblast všech trojúhelníku ve scéně - tzv. obálka. Tato obálka se vytvoří postupnou analýzou každého bodu všech trojúhelníků a zaznamenáváním maximálních a minimálních hodnot x,y,z souřadnic. Když se získají tyto minimální a maximální souřadnice, je z nich vytvořen odpovídající kvádr - obálka všech trojúhelníků.

Vytvořená obálka je použita jako hlavní uzel, který se rekurzivně dělí na osm podprostorů. Proces dělení kvádru je tvořen půlením každé hrany kvádru. Spojením všech poloviček hran se získá ohraničující oblast osmi podprostorů. Po vytyčení těchto podprostorů, je nutné určit, které trojúhelníky v nich, alespoň částečně, leží.

Nejjednodušší test zda-li trojúhelník leží v podprostoru, je zjistit jestli alespoň jeden ze tří jeho vrcholů leží uvnitř podprostoru.

Pokud ani jeden vrchol trojúhelníku neleží v podprostoru, může trojúhelník stále procházet podprostorem. Pro tento test se vytvoří rovina trojúhelníku a postupně se otestují pozice všech vrcholu kvádru, vytyčující podprostor, vůči této rovině. Pokud nejsou všechny vrcholy kvádru na jedné straně roviny trojúhelníku, trojúhelník prochází podprostorem.



Obrázek 4.2: Rozdělení prostoru ve scéně pomocí octree

Proces dělení podprostoru se rekurzivně aplikuje až do doby, kdy je buď dosažena maximální hloubka rekurze, nebo počet trojúhelníku v novém podprostoru klesne pod určitou mez. Hloubku rekurze a minimální počet trojúhelníku v podprostoru je potřeba nastavit rozumně tak, aby průchod oktalovým stromem nebyl zbytečně časově náročný.

Na obrázku 4.2 je zobrazena scéna, na které je aplikován postup dělení prostoru pomocí oktalového stromu.

4.5 Určení průsečíku paprsku s objekty scény

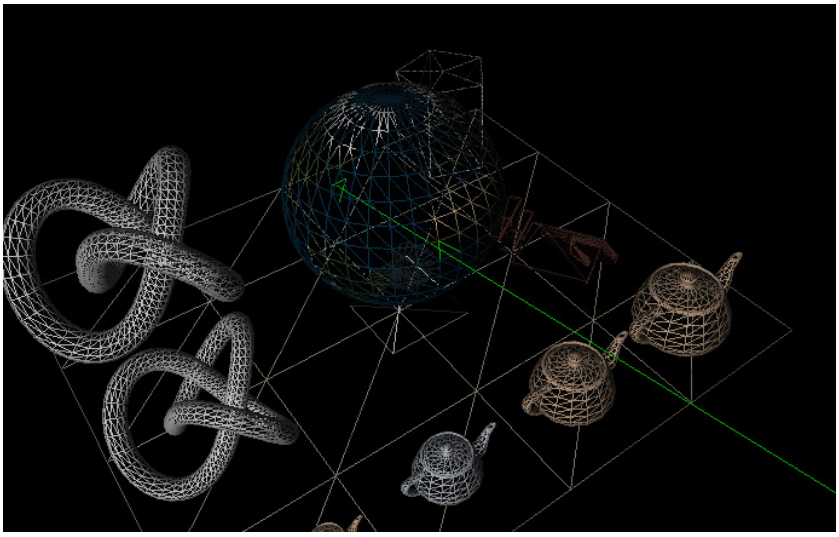
Při tvorbě raytracingu je nutné řešit průsečík paprsku s objekty ve scéně. Protože scéna je popsána pomocí trojúhelníku, je nutné vyřešit průsečík paprsku s trojúhelníkem. Pro test, zda-li paprsek protíná trojúhelník nebo nikoliv, je v programu RayTracer napsána funkce *computeTriangleIntersect*, která plně implementuje metodu, průsečíku paprsku s trojúhelníkem, popsanou v kapitole 2.7.

Pro určení trojúhelníků s kterými se paprsek protíná je nutné postupně otestovat všechny trojúhelníky ve scéně nebo podprostoru. Na obrázku 4.3 je průsečík paprsku se scénou, v které paprsek protíná pouze dva, na obrázku znázorněné, trojúhelníky. Pokud paprsek protíná více trojúhelníků, je u metoddy sledování paprsku použit ten trojúhelník, který má menší parametr t.

Pro určení průsečíku paprsku s trojúhelníkem, kdy se netestují všechny trojúhelníky ve scéně, ale pouze ty v podprostorech, slouží funkce *getIntersectOctree*.

Tato funkce nejprve otestuje průsečík paprsku se všemi podprostory (pouze na nejvyšší úrovni, nezanořuje se), prostory si uloží do prioritní fronty a vybere podprostor s nejmenším parametrem t, který rekurzivně znovu projde. Pokud, při rekurzivním průchodu, narazí na situaci, kdy podprostor neobsahuje žádné další podprostory, nachází se na nejnižší úrovní, předá tento podprostor funkci *getTriangleIntersect*.

Funkce *getTriangleIntersect* postupně testuje průsečík paprsku se všemi trojúhelníky v



Obrázek 4.3: Průsečik paprsku s trojúhelníky ve scéně.

podprostoru a jakmile projde všechny trojúhelníky, vrací ten trojúhelník, který má nejmenší parametr t. Pokud v podprostoru průsečík s trojúhelníkrm nenalezne, vrací hodnotu *false*.

Pokud se v podprostoru průsečík s trojúhelníkem nenalezne, funkce *getTriangleIntersect* se rekurzivně vrací a stejný postup aplikuje na další podprostor uložený v prioritní frontě.

4.6 Výpočet parametrů kamery

V souboru, specifikující scénu, jsou uloženy parametry pro definici kamery: pozice kamery O, vzdálenost průmětny $znear$, zorné pole (field of view - fov) fov a aspect ratio $aspect$. Pomocí těchto parametrů se vypočítá pozice kamery a průmětna, přes kterou se budou vysílat paprsky do scény.

Pro výpočet průmětny je nutné nejprve určit směr pohledu kamery $\vec{D}$. Tento směr je vypočten jako:

$$\vec{D} = P - O \tag{4.1}$$

kde P je bod v prostoru, určen v souboru popisující scénu, a O je pozice kamery.

Dále je nutné vypočítat střed průmětny S pomocí směru pohledu kamery $\vec{D}$ a vzdálenosti průmětny:

$$S = O + \vec{D}.znear \tag{4.2}$$

pro správný výpočet S je nutné aby vektor $\vec{D}$ byl normalizován - velikost vektoru byla rovna jedné.

V dalším kroku je nutné vypočítat vektor $\vec{up}$ a vektor $\vec{right}$, které jsou kolmé na směr pohledu $\vec{D}$, vypočítají se pomocí vektorového součinu jako:

$$\begin{aligned} \vec{right} &= \vec{D} \times \vec{Z} \\ \vec{up} &= \vec{right} \times \vec{D} \end{aligned} \tag{4.3}$$

kde $\vec{Z}$ je vektor s hodnotou $\vec{Z} = (0, 0, -1)$. Tyto dva vypočítané vektory a střed průmětny S udávají rovinu průmětny. Pro vysílání paprsků, je nutné znát čtyři body A, B, C, D vytyčující obdélník průmětny. Pro zjištění těchto bodů je nutné vypočítat šířku w a výšku h průmětny pomocí zorného pole a aspect ratia:

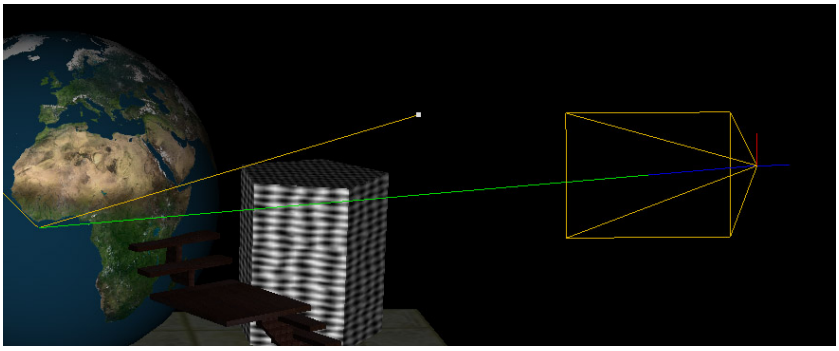
$$\begin{aligned} h &= 2.\tan\frac{fov.\frac{\pi}{180}}{2}.znear \\ w &= h.aspect \end{aligned} \tag{4.4}$$

po výpočtu šířky a výšky se vypočítají vektory $\vec{width}$ a $\vec{height}$:

$$\begin{aligned} \vec{width} &= \vec{right}.w.0.5 \\ \vec{height} &= \vec{up}.h.0.5 \end{aligned} \tag{4.5}$$

po získaní těchto vektoru, již nic nebrání výpočtu obdélníku ohraničující zobrazovací průmětnu:

$$\begin{aligned} A &= S - \vec{width} + \vec{height} \\ B &= S + \vec{width} + \vec{height} \\ C &= S - \vec{width} - \vec{height} \\ D &= S + \vec{width} - \vec{height} \end{aligned} \tag{4.6}$$



Obrázek 4.4: Zobrazená kamera, průmětna a paprsek procházející bodem S.

Pro zobrazení scény, je nutné vyslat paprsky rovnoměrně přes celou průmětnu. Docílí se toho tak, že se vypočítá vektor $\vec{AC}$, směřující zhora dolu a vektor $\vec{AB}$ směřující zleva doprava. Tyto vektory se nenormalizují. Poté se vypočítá vektor $\vec{horizontal}$ a vektor $\vec{vertical}$ pomocí:

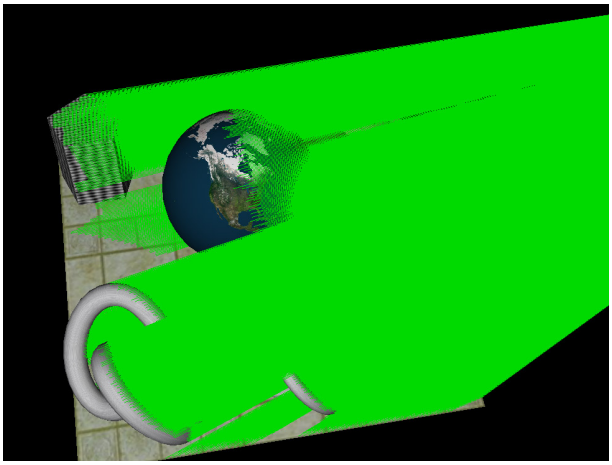
$$\begin{aligned} \vec{horizontal} &= \vec{AB}.\frac{1}{sirka} \\ \vec{vertical} &= \vec{AC}.\frac{1}{vyska} \end{aligned} \tag{4.7}$$

```
LEFT = A;
for y ← 0 to vyska do
    LOOK = LEFT;
    for x ← 0 to sirka do
        D = LOOK − O;
        raytrace(O,D, color);
        putpixel(x, y, color);
        LOOK+ = horizontal;
    end
    LEFT+ = vertical;
end
```

Algorithm 1: Rovnoměrné vysílání paprsků přes průmětnu.

kde konstanta *sirka* a *vyska* jsou hodnoty udávající rozlišení výsledného obrazu a vektory $\vec{horizontal}$ a $\vec{vertical}$ jsou poté přírůstky vektoru v daném směru. Jejich postupným přičítáním se docílí rovnoměrného vysílání paprsku přes průmětnu.

Algoritmus 1 popisuje postup, kterým se docílí rovnoměrného vyslání paprsků přes průmětnu. V tomto algoritmu je znázorněn postup určení barvy obrazu a uložení barvy do framebufferu na patřičnou pozici.



Obrázek 4.5: Rovnoměrné vysílání paprsků do scény.

Na obrázku 4.5 jsou znázorněny paprsky, které jsou vysílány přes průmětnu do scény. Zobrazeny jsou pouze paprsky, které směřují pouze do scény, mají průsečík s tělesem ve scéně.

V této podkapitole byly popsány postupy, které byly použity v programu RayTracer, pro vytvoření zobrazovací průmětny a postup pro rovnoměrné vysílání paprsků přes zobrazovací průmětnu.

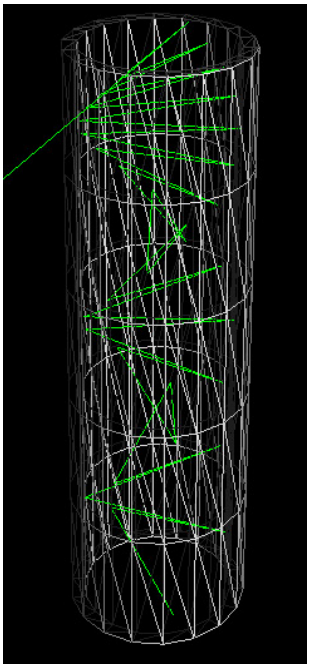
4.7 Sledování paprsku

Pro vypočítání výsledného obrazu je nutné určit barvu každého paprsku, který se vysílá skrze průmětnu směrem do scény. Postup, který je použit v programu RayTracer, se snaží implementovat algoritmus sledování paprsku uvedený v první části práce - 2.6.1, s tím rozdílem, že vyhodnocení osvětlovacího modelu v daném bodě se vypočítává až po vyslání odraženého paprsku.

Po nalezení průsečíku paprsku s objekty ve scéně je nutné, pro sledování paprsku scénou, spočítat směr odraženého paprsku. Odražený paprsek $\vec{R}$ se počítá jako:

$$\vec{R} = \vec{V} - 2.\vec{N}.(\vec{V}.\vec{N}) \tag{4.8}$$

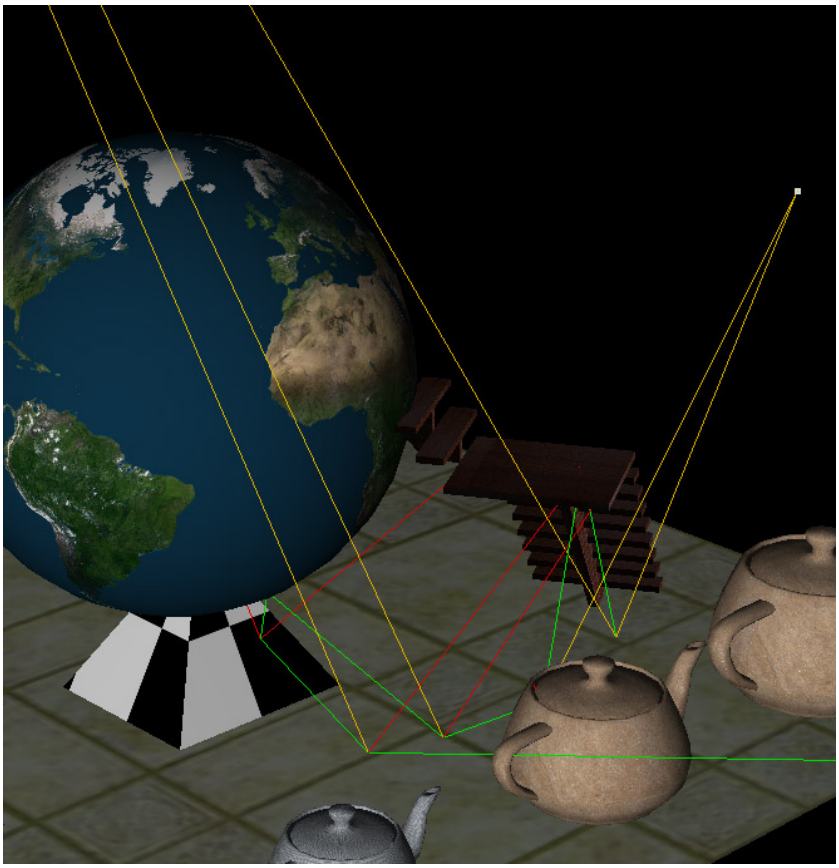
kde $\vec{V}$ je příchozí paprsek a $\vec{N}$ je normála povrchu v bodě odraženého paprsku.



Obrázek 4.6: Paprsky, které se odráží skrze dutou trubici.

Výpočet odraženého paprsku je implementován ve funkci *reflectVector*. Na obrázku 4.6 je scéna, která byla vytvořena pro testování odrazu paprsků a na které se testuje, jak se paprsky odráží od vnitřní stěny dutého válce.

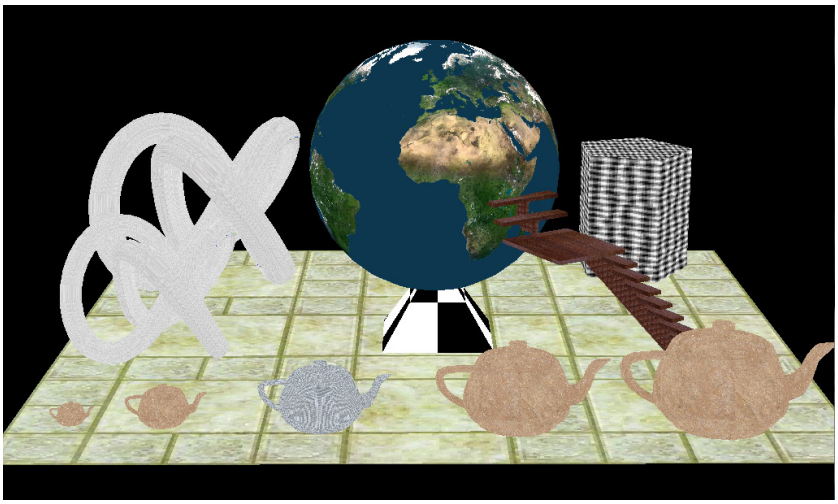
Paprsek, pro který se vyhodnocuje barva, se vyšle přes průmětnu směrem do scény, kde se hledá průsečík se scénou. Po nalezení průsečíku se vyšlou stínové paprsky ke každému světelnému zdroji, aby se zjistilo zda-li je světelný zdroj zakrytý či nikoliv. Dále se spočítá odražený paprsek, průsečík paprsku se určí jako počátek paprsku a postup se rekurzivně opakuje.



Obrázek 4.7: Sledování jednoho paprsku.

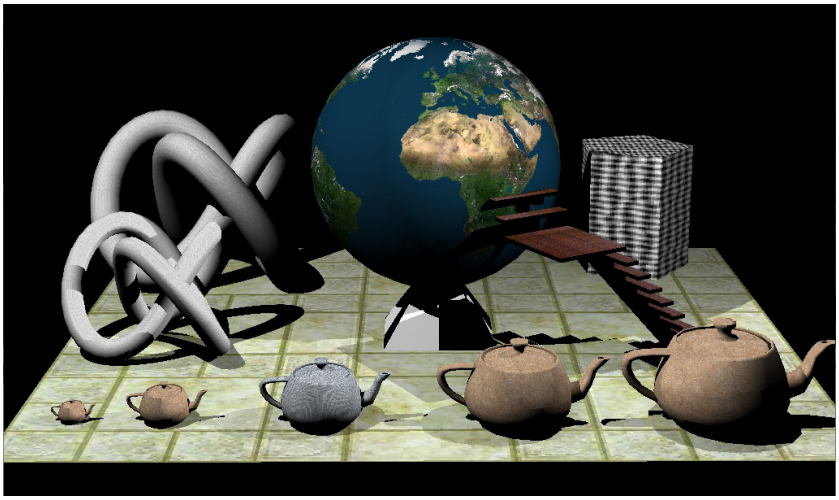
Tento postup popisuje obrázek 4.7, na kterém jsou zobrazeny paprsky použité pro výpočet barvy jednoho pixelu výsledného obrazu: zelenou barvou jsou zobrazeny primární a sekundární paprsky, žlutou barvou jsou zobrazeny paprsky stínové, které nejsou v zákrytu s žádným objektem ve scéně a červené paprsky jsou stínové paprsky, které mají mezi průsečíkem a světelným zdrojem nějaké stínící těleso nebo objekt.

Protože metoda sledování paprsku je rekurzivní algoritmus - pro výpočet finální barvy potřebuje znát barvu odraženého paprsku - bylo, při tvorbě programu, nutné nejprve vyřešit výpočet osvětlovacího modelu pouze pro primární paprsek a až poté rekurzivně vytvářet sekundární paprsky a přičítat je do osvětlovacího modelu.



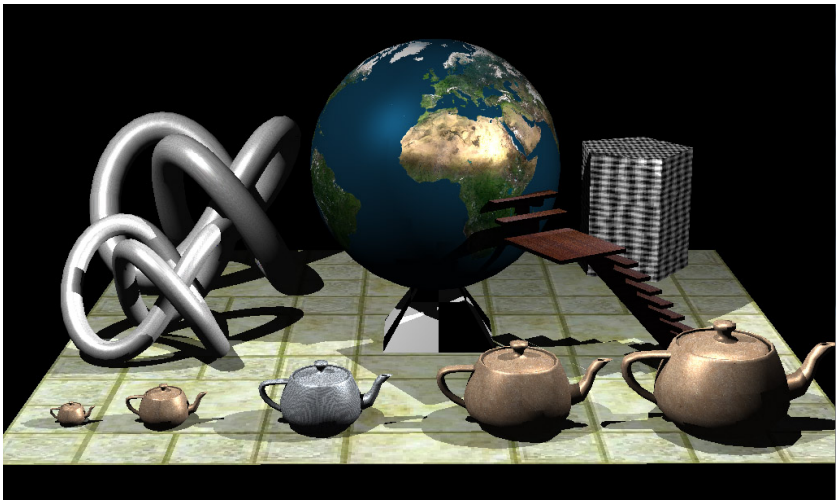
Obrázek 4.8: Scéna zobrazena pomocí konstantního stínování.

Nejprve (4.8) byla, pro výpočet osvětlovacího modelu, využita pouze difúzní složka světla, bez jakýchkoliv úprav a byly ignorovány stínové paprsky. Hodnota difúzní složky byla určena z textury. Pro zjištění barvy z textury je nutné vypočíst uv souřadnice v průsečíku, ty se určí interpolováním uv souřadnic vrcholů trojúhelníku pomocí barycentrických souřadnic, získaných při výpočtu průsečíku paprsku s trojúhelníkem.



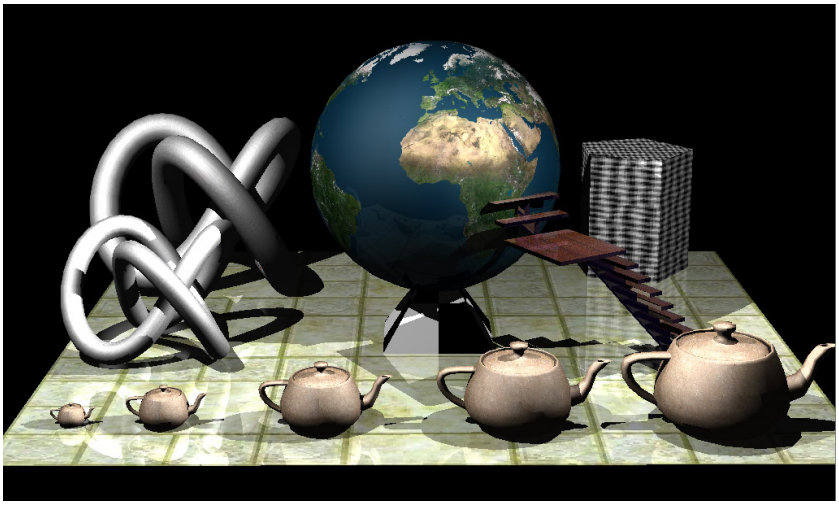
Obrázek 4.9: Difúzní složka světla.

V dalším kroku (4.9) byly do výpočtu zahrnuty stínové paprsky. Difúzní složka byla připočtena do osvětlovacího modelu pouze tehdy, když stínový paprsek nenalezl žádný objekt mezi průsečíkem a světelným zdrojem, navíc byla difúzní složka vynásobena skalárním součinem mezi normálou povrchu v průsečíku a směrem stínového paprsku.



Obrázek 4.10: Difúzní, odrazivá a ambientní složka světla.

V následujícím kroku (4.10) je do osvětlovacího modelu připočtena ambientní a odrazivá složka světla. Ambientní složka světla je připočítána do osvětlovacího modelu vždy a zajišťuje částečné osvětlení míst na které nedopadá světlo ani z jednoho světelného zdroje. Odrazivá složka je připočítána, podobně jako difúzní složka, pouze když průsečík není ve stínu světelného zdroje. Hodnota barvy odrazivé složky je určena ze specifikace modelu (nejčastěji čistě bílá) a je vynásobena skalárním součinem mezi odraženým světelným paprskem a paprskem k pozorovateli.

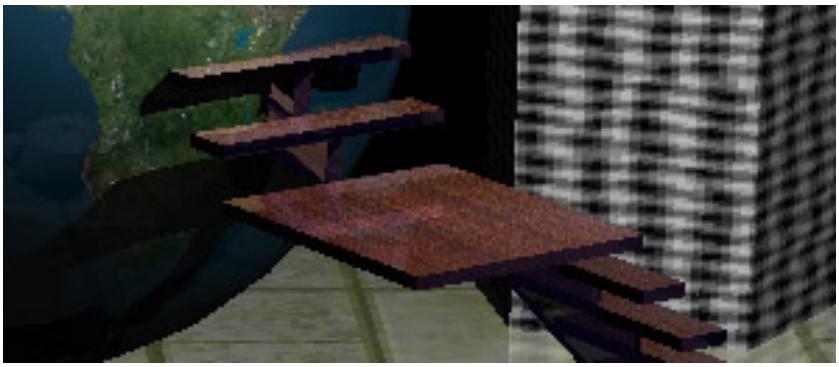


Obrázek 4.11: Scéna zobrazena pomocí metody sledování paprsku.

Protože v předchozím kroku byl Phongův osvětlovací model plně vypočítán, jsou ve finálním kroku (4.11) do výpočtu osvětlovacího modelu zahrnuty rekurzivně vytvořené sekundární paprsky, čímž se zobrazila scéna pomocí metody sledování paprsku. Problém v této scéně je v tzv. aliasech, které se bude snažit řešit následující kapitola o AntiAliasingu.

4.8 AntiAliasing

Problém scén, vykreslených pomocí metody sledování paprsku, je ten, že jsou hodně ovlivněné tzv. aliasem. Problém aliasu nastává, protože paprsky, které udávají barvu pixelu, jsou vyslány středem pixelu na průmětně a jsou velmi tenké. Barvu pixelu tedy udává malá oblast scény, na kterou dopadne paprsek, místo toho, aby barva byla určena z oblasti scény, který jeden pixel popisuje. Vysláním jednoho paprsku na pixel, bez jakýchkoliv úprav, vznikají ve výsledné scéně zubaté hrany a aliasy v texturách.



Obrázek 4.12: Zubaté hrany a aliasy v textuře.

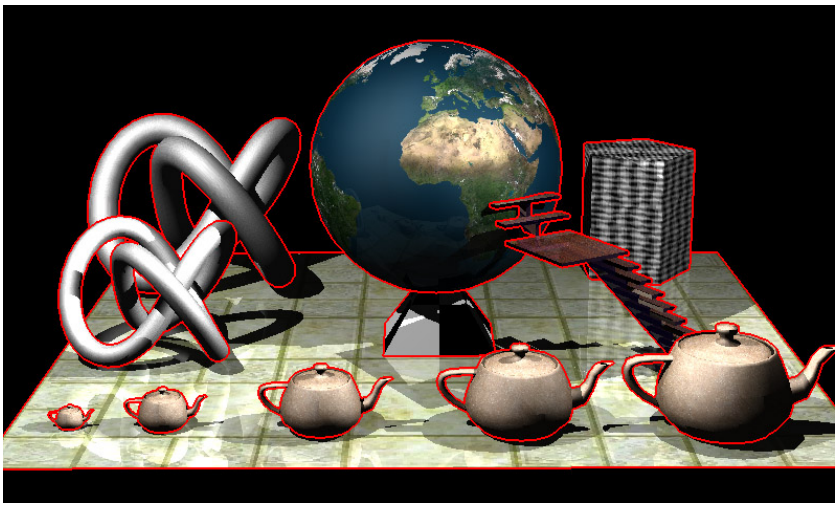
Jedním z možných řešení aliasu je použití metody supersamplingu (česky nadvzorkování), kdy se pro každý pixel obrazu vyšle více - N - paprsků. Výsledná barva pixelu je poté určena jako průměr barvy N paprsků.

Obrázek 4.13: Detail vykreslené scény s použitím supersamplingu.

Na obrázku 4.13 lze vidět, jak metoda supersamplingu odstranila aliasy z textury dřeva a jak vyhladila hrany objektů. V programu RayTracer byl využit supersampling, kdy pro hodnotu jednoho pixelu bylo vyhodnoceno 9 paprsků. Bylo toho dosáhnuto tak, že se vypočítala scéna s devětkrát větším rozlišením (třikrát větší na šířku, třikrát větší na délku) a výsledná scéna byla zmenšena tím, že se jeden pixel počítal jako průměr devíti pixelů.

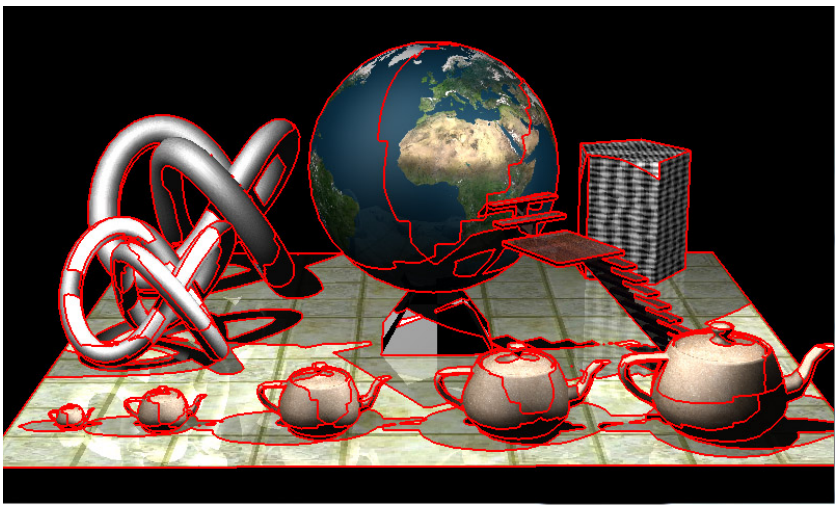
Problém metody supersamplingu je ten, že má velkou časovou náročnost, protože je nutné počítat pro každý pixel scény N paprsků navíc. Lepší řešení je využití mipmapingu pro textury a pro hrany objektů využít nadvzorkování.

Nadvzorkování přechodů mezi objekty



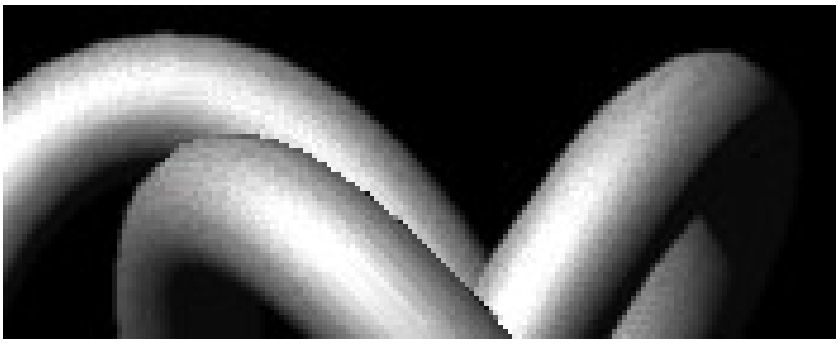
Obrázek 4.14: Znázornění nadvzorkovaných oblastí v přechodech mezi objekty.

Pro určení nadvzorkovaných oblastí ve scéně, se mírně upraví algoritmus postupného vysílání paprsku přes průmětnu 1. Pro toto určení je vytvořeno pole struktur *aagrid* o stejné velikosti jako framebuffer, do kterého se zapisují zobrazené objekty na x, y souřadnicích. Před zapsáním výsledné barvy do framebufferu, se zjistí, zda-li body na pozicích $x - 1$ nebo $y - 1$ byly vykresleny ze stejného objektu. Pokud ano, barva paprsku je zapsána do framebufferu, pokud ne, vytvoří se dalších osm paprsků (kolem již vyslaného paprsku), z kterých se vypočítá průměrná barva a uloží se do framebufferu. Při nadvzorkování pixelu na souřadnicích x, y je nutné také nadvzorkovat a upravit barvu pixelu na pozici $x-1$ nebo $y - 1$ podle toho, kde nesousedí stejné objekty. Znázorněné body výsledného obrazu, které je nutné nadvzorkovat, jsou zobrazeny na obrázku 4.14.



Obrázek 4.15: Znázornění nadvzorkovaných oblastí v přechodech mezi objekty a stíny.

Předchozí postup zajistil odstranění zubatých hran objektů. Ve scéně ovšem zůstávají zubaté přechody mezi přechodem mezi stínem a osvětlenou části objektu. Pro určení těchto oblastí, by bylo možné rozšířit algoritmus uvedený výše, o kontrolu světel. Problém při implementaci však nastane při určení počtu světel, kdy není dopředu známo kolik světel ve scéně bude. Proto byl algoritmus upraven, kdy každému objektu a každému světlu je přiřazen číselný identifikátor - příznak, při průchodu paprsku se do kontrolního součtu přičítají příznaky objektu a světel, které na bod dosvítí. Kontrola mezi sousedními pixely se pak odehrává na rovnosti kontrolních součtů.

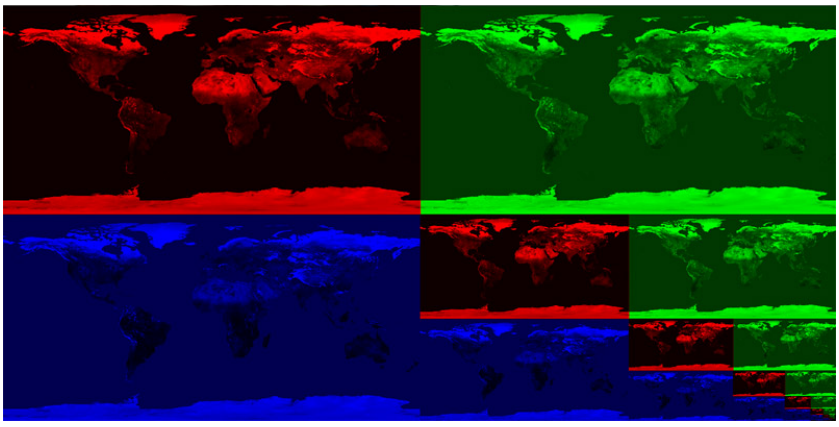


Obrázek 4.16: Zubaté hrany v rámci jednoho objektu.

Poslední problém zubatých hran nastává u hran v rámci jednoho objektu, jak je znázorněno na obrázku 4.16. Odstranění těchto hran se dá docílit tím, že se do kontrolního součtu budou přičítat příznaky trojúhelníku místo příznaků objektu. I když tento postup vyhladí hrany v rámci jednoho objektu, není v programu RayTracer využit, protože způsobí nadvzorkování na každém přechodu mezi trojúhelníky, čímž zbytečně zvýší časovou náročnost. Další řešení tohoto problému může být rozdělením objektů tak, aby překrývající části byli nové objekty.

Antialiasing textur - MIP map

MIP (z latinského multum in parvo) map je způsob mapování textury, kdy se při načtení textury do paměti předpočítají i její zmenšené verze. Při určení barvy z textury se poté vybírá verze textury v závislosti na vzdálenosti průsečíku od kamery.



Obrázek 4.17: Způsob uložení mipmapy.

Textura poté není uložená v poli rgb hodnot, ale v poli char hodnot, kdy pro každou barvu je vymezen určitý úsek. Uložení textury je znázorněno na obrázku 4.17.

Pro určení barvy jednoho texelu je nutné mírně upravit mapování, kdy je nutné pro každou složku výsledné barvy se odkázat do správného úseku a taky do správné hloubky.

Kapitola 5

Závěr

Cílem této práce bylo naimplementovat fotorealistickou metodu zobrazení scény se zaměřením na metodu sledování paprsku. Tato práce postupně popisuje postupy a milníky práce, které vedly k úspěšnému splnění tohoto cíle.

Prostudované metody realistického zobrazování jsou popsány v první částí práce, kde jsou popsány metody pro fotorealistické zobrazení scény a je zde popsán software pro realistické zobrazení. Vhodná metoda pro implementaci byla vybrána metoda sledování paprsku a model dat pro reprezentaci realisticky zobrazované scény byl navrhnut v třetí části práce, kdy pro načtení souboru specifikující scénu bylo použito aplikačního rozhraní COLLADA DOM. V třetí části práce je popsán postup, který vedl k implementaci softwaru pro zobrazení scény pomocí metody sledování paprsku. Je zde také demonstrována funkčnost této metody.

Prací na téhle práci jsem získal cenné zkušenosti z oblasti fotorealistické počítačové grafiky. Tato práce mě naučila mnoho principů a metod, které se v dnešní počítačové grafice používají.

V práci bych rád pokračoval tak, že bych chtěl implementovat metodu photon mappingu, kdy bych načtení scény a model pro reprezentaci scény použil z této práce. Dále by se dal využít způsob výpočtu parametru kamery, průsečík paprsku s trojúhelníkem a průchod paprsku scénou. Rozdělení podprostoru z této práce nebude pravděpodobně použito, protože pro photon mapping je doporučené rozdělení podprostoru pomocí kd stromu.

Literatura

[1] Appel, A.: Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, AFIPS '68 (Spring), New York, NY, USA: ACM, 1968, s. 37–45, doi:10.1145/1468075.1468082.

[2] Ashikhmin, M.; Shirley, P.: An Anisotropic Phong BRDF Model. *Journal of Graphics Tools*, ročník 5, 2000: s. 25–32.

[3] Cohen, M. F.; Chen, S. E.; Wallace, J. R.; aj.: A progressive refinement approach to fast radiosity image generation. *SIGGRAPH Comput. Graph.*, ročník 22, č. 4, Červen 1988: s. 75–84, ISSN 0097-8930, doi:10.1145/378456.378487.

[4] Cook, R. L.; Porter, T.; Carpenter, L.: Distributed ray tracing. *SIGGRAPH Comput. Graph.*, ročník 18, č. 3, Leden 1984: s. 137–145, ISSN 0097-8930, doi:10.1145/964965.808590.

[5] Foley, J.: *Computer Graphics: Principles and Practice, Second Edition in C*. Addison-Wesley systems programming series, Addison Wesley Professional, 1996, ISBN 9780201848403.

[6] Foundation, B.: Blender Features. 2012, [Online; accessed 11-May-2013]. URL http://www.blender.org/features-gallery/features/

[7] Goral, C. M.; Torrance, K. E.; Greenberg, D. P.; aj.: Modeling the interaction of light between diffuse surfaces. *SIGGRAPH Comput. Graph.*, ročník 18, č. 3, Leden 1984: s. 213–222, ISSN 0097-8930, doi:10.1145/964965.808601.

[8] Gouraud, H.: Continuous Shading of Curved Surfaces. *IEEE Trans. Comput.*, ročník 20, č. 6, Červen 1971: s. 623–629, ISSN 0018-9340, doi:10.1109/T-C.1971.223313.

[9] Havel, J.; Herout, A.: Yet Faster Ray-Triangle Intersection (Using SSE4). *IEEE Transactions on Visualization and Computer Graphics*, ročník 16, č. 3, 2010: s. 434–438, ISSN 1077-2626.

[10] Jensen, H. W.: Global illumination using photon maps. In *Proceedings of the eurographics workshop on Rendering techniques '96*, London, UK, UK: Springer-Verlag, 1996, ISBN 3-211-82883-4, s. 21–30.

[11] Jensen, H. W.; Christensen, N. J.: Efficiently Rendering Shadows Using The Photon Map. In *Edugraphics + Compugraphics Proceedings*, Publications, 1995, s. 285–291.

[12] Jiří Žára, J. S. a. P. F., Bedřich Beneš: *Moderní počítačová grafika*. Computer Press, 2004, iSBN 80-251-0454-0.

[13] Kajiya, J. T.: The rendering equation. *SIGGRAPH Comput. Graph.*, ročník 20, č. 4, Srpen 1986: s. 143–150, ISSN 0097-8930, doi:10.1145/15886.15902.

[14] Nicodemus, F. E.: Directional reflectance and emissivity of an opaque surface. *Applied Optics*, ročník 4, č. 7, 1965: s. 767–775.

[15] Phong, B. T.: Illumination for computer generated pictures. *Commun. ACM*, ročník 18, č. 6, Červen 1975: s. 311–317, ISSN 0001-0782, doi:10.1145/360825.360839.

[16] Trevisan, C.: Rendering 03. 2002, [Online; accessed 11-May-2013].
URL
http://www.camillotrevisan.it/cad2002/rendering_03/Rendering_03.htm

[17] Watt, A.; Watt, M.: *Advanced animation and rendering techniques*. New York, NY, USA: ACM, 1991, ISBN 0-201-54412-1.

[18] Whitted, T.: An improved illumination model for shaded display. *Commun. ACM*, ročník 23, č. 6, Červen 1980: s. 343–349, ISSN 0001-0782, doi:10.1145/358876.358882.

[19] Wikipedia: Autodesk 3ds Max. 2013, [Online; accessed 11-May-2013].
URL http://en.wikipedia.org/wiki/Autodesk_3ds_Max

[20] Wikipedia: Blender (software). 2013, [Online; accessed 11-May-2013].
URL http://en.wikipedia.org/wiki/Blender_%28software%29

[21] Wikipedia: Blinn?Phong shading model. 2013, [Online; accessed 11-May-2013].
URL http://en.wikipedia.org/wiki/Blinn%E2%80%93Phong_shading_model

[22] Wikipedia: Phongův osvětlovací model. 2013, [Online; accessed 11-May-2013].
URL
http://cs.wikipedia.org/wiki/Phong%C5%AFv_osv%C4%9Btlovac%C3%AD_model

[23] Wikipedia: POV-Ray. 2013, [Online; accessed 11-May-2013].
URL http://en.wikipedia.org/wiki/POV-Ray

[24] Wikipedia: Radiosity (computer graphics). 2013, [Online; accessed 11-May-2013].
URL http://en.wikipedia.org/wiki/Radiosity_%28computer_graphics%29

[25] Wikipedia: Ray tracing (graphics). 2013, [Online; accessed 11-May-2013].
URL http://en.wikipedia.org/wiki/Ray_tracing_%28graphics%29

[26] Wikipedia: Śledzenie promieni. 2013, [Online; accessed 11-May-2013].
URL http://pl.wikipedia.org/wiki/%C5%9Aledzenie_promieni

Příloha A

Obsah CD

- spustitelný program
- zdrojové soubory k této práci
- obrázky, které byly vytvářeny v průběhu práce