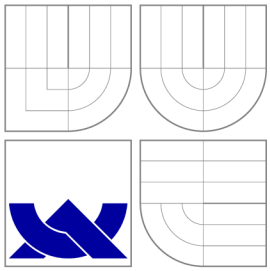
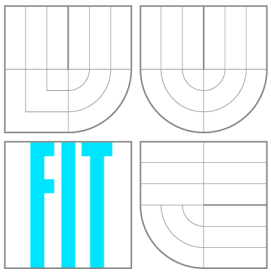


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VÝZKUMNÉ CENTRUM INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY
RESEARCH CENTRE OF INFORMATION TECHNOLOGY

PHYSICALLY BASED SHADING

PHYSICALLY BASED SHADING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ŠIMON MATĚJ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. TOMÁŠ STARKA

BRNO 2016

Zadání bakalářské práce/16903/2015/xmatej45

Vysoké učení technické v Brně - Fakulta informačních technologií

Výzkumné centrum informačních technologií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Matěj Šimon**
Obor: Informační technologie
Téma: **Physically Based Shading**
Physically Based Shading
Kategorie: Počítačová grafika

Pokyny:

1. Nastudujte metodu zvanou Physicaly Based Shading.
2. Nastudujte OpenGL API.
3. Navrhněte aplikaci demonstrující metodu Physicaly Based Shading.
4. Implementujte demonstrační aplikaci.
5. Vytvořte plakát nebo video prezentující práci.

Literatura:

- J Bainbridge: Physically Based Shading, http://www.joshbainbridge.com/files/rendering_report.pdf
- Stephen McAuley, Stephen Hill, Naty Hoffman, Yoshiharu Gotanda, Brian Smits, Brent Burley, and Adam Martinez. 2012. Practical physically-based shading in film and game production. In *ACM SIGGRAPH 2012 Courses* (SIGGRAPH '12). ACM, New York, NY, USA, , Article 10 , 7 pages. DOI=http://dx.doi.org/10.1145/2343483.2343493
- M Pharr, G Humphreys: Physically based rendering: From theory to implementation

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Starka Tomáš, Ing.**, VCIT FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Výzkumné centrum informačních technologií
L.S.
Božetěchova 2, 612 66 Brno

prof. Ing. Tomáš Hruška, CSc.
vedoucí ústavu

Abstrakt

Tato práce se zabývá zobrazováním materiálů v počítačové grafice, za použití metod, vycházejících z fyzikálních vlastností těchto materiálů. Jsou zde rozebrány fyzikální principy šíření světla a jeho interakce s objekty. Dále jsou popsané metody pro zobrazování, vycházející z těchto principů, a jejich vhodnost pro implementaci. Tyto metody jsou pak demonstrovány v aplikaci.

Abstract

This thesis deals with imaging of materials in computer graphics, using rendering methods based on physical properties of these materials. Physical principles of light propagation and its interactions with objects are analyzed, several rendering methods based on these principles are described and then analyzed for suitability in implementations. Some of these methods are demonstrated in application.

Klíčová slova

PBS, shadery, real-time, osvětlení, OpenGL, Assimp, Qt

Keywords

PBS, shaders, real-time, lighting, OpenGL, Assimp, Qt

Citace

MATĚJ, Šimon. *Physically Based Shading.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Starka Tomáš.

Physically Based Shading

Prohlášení

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

.....
Šimon Matěj
18. května 2016

Poděkování

Děkuji vedoucímu Ing. Tomášovi Starkovi za pomoc a ochotu při tvorbě práce a konzultacích.

Obsah

Kapitola 1

Úvod

S neustále se zvyšujícím výkonem počítačů se vizuální stránka videoher či jiných grafických aplikací přibližuje realitě. V grafických enginech se proto stále častěji objevují systémy, umožňující definici materiálů na základě jejich fyzikálních vlastností. Narozdíl od systému, kde se za pomocí atributů či textur dosahuje přijatelného vzhledu v pouze určitých světelných podmínkách, fyzikální přístup umožňuje použít jednu definici pro všechny případy.

Aby bylo možné něčeho takového dosáhnout, je nutné důkladně popsat vlastnosti materiálů a implementovat metody, které budou s takovými materiály korektně pracovat. Těmito vlastnostmi jsou kupříkladu indexy odrazivosti a refrakce.

Speciálním typem materiálů jsou kovy, u nichž je jediným viditelným projevem odrazivost. Proto je třeba pro jejich zobrazení odrazivost vykreslovat, což je ovšem výpočetně náročné. Tato práce se proto zaměřuje na její korektní zobrazení, s ohledem na vykreslování v reálném čase.

V práci je dále popsána fyzikální podstata těchto vlastností, analyzovány určité zobrazovací metody a je vytvořen systém pro definování vlastností materiálů. Ten je poté implementován v aplikaci, která jej využívá a demonstruje popsané metody.

Kapitola 2

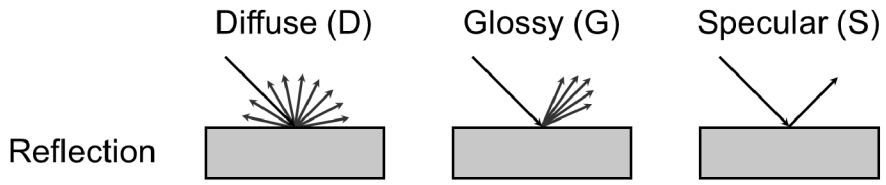
Teorie

2.1 Definice PBS

Physically Based Shading označuje zobrazovací metody v počítačové grafice, které se snaží simulovat interakci materiálů a světla, která nastává v reálném světě. Účelem je dosažení systému snadné definice materiálů, které budou simulovat chování jejich reálných předobrazů v jakýchkoliv světelných podmínkách.

2.2 Odrazivost

Odrazivost je optická vlastnost materiálu udávající míru odraženého světla od jeho povrchu. V závislosti na povrchu materiálu se odrazivost dělí na rozptýlenou a přímou. Rozptýlená odrazivost se vyskytuje u matných neboli difůzních povrchů, kdy se záření odráží různými směry a proniká pod povrch materiálu a zpět. Takto se jeví pozorovateli jako jednolitá barva. Takové povrchy se nazývají Lambertovské [2]. Přímá neboli spekulární odrazivost vzniká na hladkém povrchu, kde odražené světlo tvoří obraz odrážených předmětů. Většina reálných materiálů (kromě kovů) kombinuje oba typy odrazivosti. Za speciální typ přímé odrazivosti může být považován odlesk, který vzniká v případech, kdy povrch materiálu není úplně hladký a tak je odraz rozmazaný 2.1.



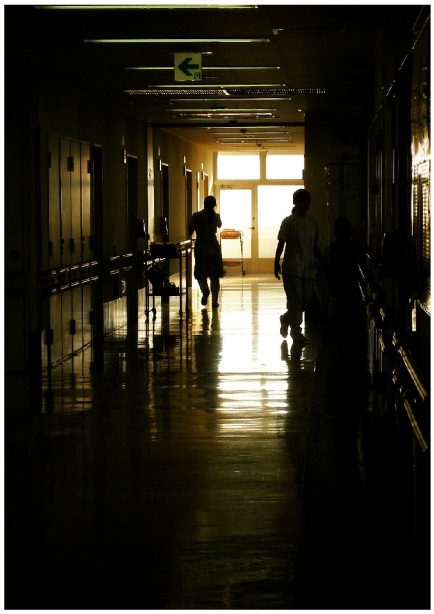
Obrázek 2.1: Rozdíly v odrazu světla – difúze, odlesk, spekulární odraz.

V počítačové grafice se přístupy pro zobrazování přímé a rozptýlené odrazivosti liší. Pokud dokonale matné povrchy odrážejí světlo téměř všemi směry, pak každý bod materiálu odráží světlo z téměř všech směrů. Fyzikálně správný přístup, pro určení výsledné barvy v jednom bodě materiálu, by pak požadoval výpočet na základě všech zdrojů světla ze všech směrů, odkud může světlo na daný bod dopadat. Jelikož by takový výpočet byl velmi složitý, v počítačové grafice se buďto zjednodušuje (například pomocí metody Monte

Carlo [1]), nebo nahrazuje jinými výpočty, produkujícími podobné výsledky. Pro homogenní materiály je míra odrazivosti dána fresnelovým koeficientem, který udává poměr odraženého ku dopadajícímu světlu [4].

2.2.1 Fresnel

Optický jev, nazývaný *fresnel* je pojmenován podle francouzského fyzika Augustina-Jeana Fresnela. Dopad tohoto jevu je velmi viditelný na málo odrazivých materiálech – i takové mají silný odraz v případech, kdy se úhel dopadajícího světla blíží nule 2.2.



Obrázek 2.2: Fresnel – při velmi nízkém úhlu dopadajícího světla se vlastnosti, jinak málo odrazivé podlahy, blíží zrcadlu.

Fresnelovy rovnice popisují chování světla při pohybu mezi médii s různými indexy lomu. Pokud světlo prochází mezi dvěma takovými materiály, může nastat jak lom světla, tak i jeho odraz. Fresnelovy rovnice pak udávají poměr odraženého a lomeného světla a také popisují fázový posun takto odraženého světla [4].

Pro výpočet vlivu *fresnelova faktoru* při spekulárním odrazu se počítačové 3D grafice využívá Schlickovy aproximační rovnice 2.1 [7].

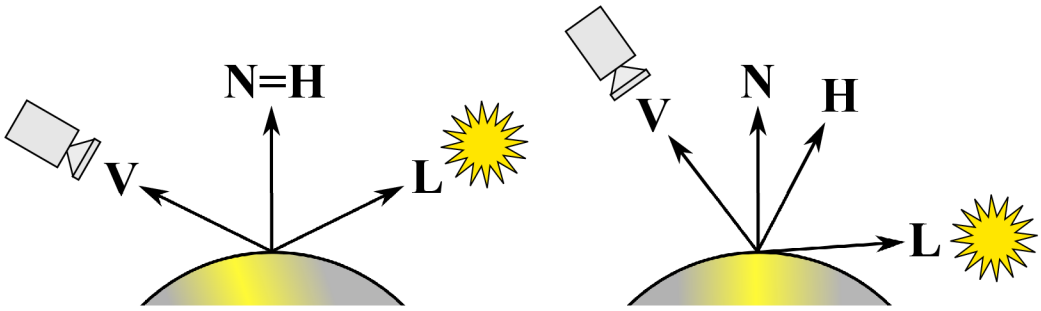
$$\begin{aligned} R(\theta) &= R_0 + (1 - R_0)(1 - cos\theta)^5 \\ [H] \quad R_0 &= \left(\frac{n_1 - n_2}{n_1 - n_2}\right)^2 \end{aligned} \tag{2.1}$$

Rovnice může být přepsána jako 2.2 pro f_λ, která udává odrazivost materiálu pro danou vlnovou délku. Počítání koeficientu fresnelu pro různé vlnové délky zvlášť, je důležité především u kovových materiálů, protože právě různá odrazivost pro různé vlnové délky světla dává kovovým materiálům jejich odstín [5].

$$
\begin{aligned}
F_\lambda &= f_\lambda + (1 - f_\lambda)(1 - \mathbf{H} \cdot \mathbf{V})^5 \\
\mathbf{H} &= \left(\frac{\mathbf{V} \times \mathbf{L}}{|\mathbf{V} \times \mathbf{L}|} \right)
\end{aligned}
\tag{2.2}
$$

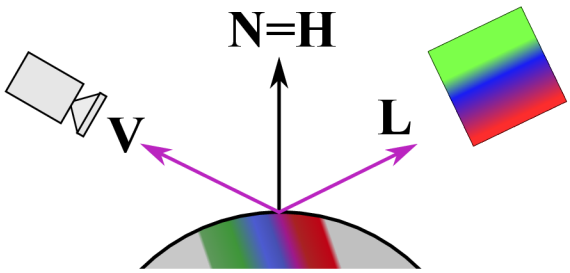
kde: **V** – vektor k pozorovateli
L – vektor ke zdroji světla
H – half vektor

Při nasvětlení objektu bodovým zdrojem světla je Schlickova aproximace Fresnelova faktoru závislá na vektoru k pozorovateli **V** a ke zdroji světla **L**, half vektor **H**, který leží mezi vektory **V** a **L** se může lišit od normálového vektoru **N** 2.3.



Obrázek 2.3: Fresnel – při výpočtu koeficientu pro osvětlení bodovým světlem nezáleží na normálovém vektoru.

V případě použití Fresnelova faktoru pro přímou odrazivost zůstává vektor k pozorovateli stejný, ale vektor ke zdroji světla se v takovém případě vypočítá podle úhlu odrazu paprsku. Jelikož se úhel dopadu paprsku rovná úhlu jeho odrazu, je úhel mezi vektorem k pozorovateli **V** a normálovým vektorem **N** roven úhlu mezi vektorem k pozorovateli **L** a normálovým vektorem **N**. V takovém případě se tedy half vektor **H** rovná normálovému vektoru, jak je ukázáno na obrazku 2.4.



Obrázek 2.4: Fresnel - při počítání fresnelu pro přímou odrazivost je normálový vektor roven half vektoru.

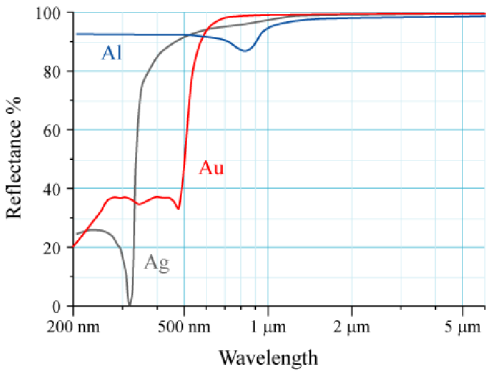
2.2.2 Kovové a nekovové materiály

U nekovových materiálů se i v případě hladkého povrchu světlo rozptyluje. Barva materiálu, kterou vnímáme, je u nekovových materiálů dána zejména difúzí. Při přímé odrazivosti totiž

většina nekovových materiálů odráží všechny vlnové délky světla přibližně stejně intenzivně, takže při pozorování v odrazu žádný barevný nádech nevnímáme. U kovů se, narozdíl od nekovových materiálů, difúze (rozptyl světla) neprojevuje. Pokud je kovový povrch hladký, veškeré světlo vycházející z materiálu je výsledkem přímé odrazivosti. Výrazné zabarvení některých kovů (viz 2.5) je dáno různou intenzitou odráženého světla různých vlnových délek [4]. Například u mědi je její načervenalá barva důsledkem vyšší odrazivosti pro nižší vlnové délky viditelného světla. Ukázka naměřených hodnot odrazivosti pro různé druhy kovů je na obrázku 2.6.

V praxi se na površích kovů často vyskytují vrstvy jiných materiálů, jako třeba nečistoty nebo koroze, které pak mění vlastnosti povrchu.

Obrázek 2.5: Ilustrace zabarvení některých druhů kovů - stříbro, zlato, měď. [google]



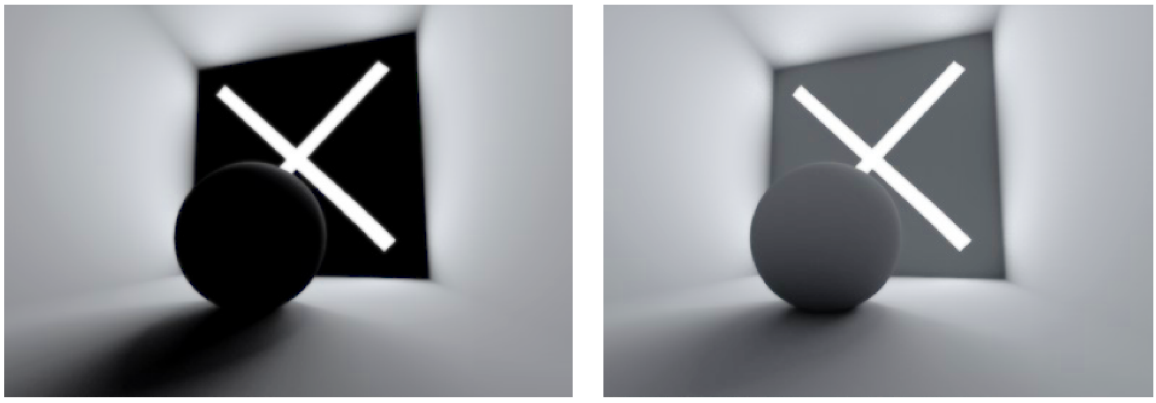
Obrázek 2.6: Naměřené hodnoty odrazivosti záření různých vlnových délek u některých kovů. [photonics.com]

Míra odrazivosti u materiálu vychází z jeho indexu lomu. Tento index má svou reálnou a imaginární část $(n + kj)$. U většiny nevodivých materiálů se imaginární část blíží nule, proto se dá míra odrazu spočítat velmi přesně i s jejím zanedbáním (2.1). U kovů bývá ovšem hodnota imaginární části indexu vyšší, a tedy i významnější pro výpočet. Schlickova aproximace počítá pouze s reálnou složkou, je proto určena pro nevodivé materiály a nikoli pro výpočet odrazu na kovovém povrchu. K určení Fresnelova faktoru pro kovové materiály je pak potřeba výpočtu, který s touto složkou počítá, například 2.3 [5].

$$F(\theta) = \frac{(n-1)^2 + 4n(1-\cos\theta)^5 + k^2}{(n+1)^2 + k^2} \tag{2.3}$$

2.3 Osvětlení

Světelné záření v reálném světě vzniká při žhavosti, což je fyzikální jev, kdy zahřáté těleso emituje záření, nebo jako výsledek některých chemických (fluorescence) či fyzikálních (v polovodičích) procesů. Z hlediska simulace jeho vlivu je ale důležitější způsob jeho šíření, který záleží na tom, co je zdrojem světla. Zdroje světla se dají rozdělit na přímé a nepřímé. Přímé zdroje světla jsou například slunce, nebo žárovka, nepřímým zdrojem světla je pak veškeré odražené či lomené světlo z okolního prostředí 2.7. Ačkoli se z fyzikálního hlediska světlo chová stejně, nehledě na jeho zdroj, v počítačové grafice se ke světlu z různých zdrojů přistupuje odlišně.



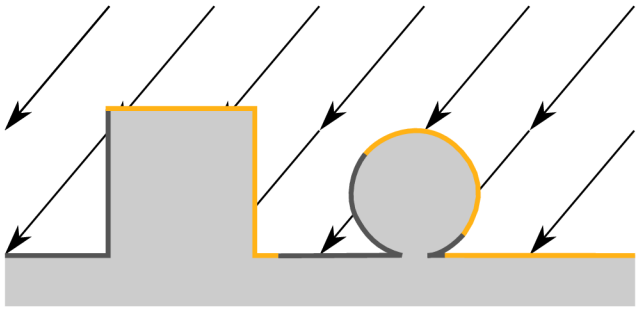
Obrázek 2.7: Rozdíl v nasvícení pouze přímým zdrojem světla (vlevo) a kombinací přímého a nepřímého osvětlení (vpravo).

2.3.1 Přímé zdroje světla

Přímé zdroje světla se mohou rozdělit podle pozice na několik typů:

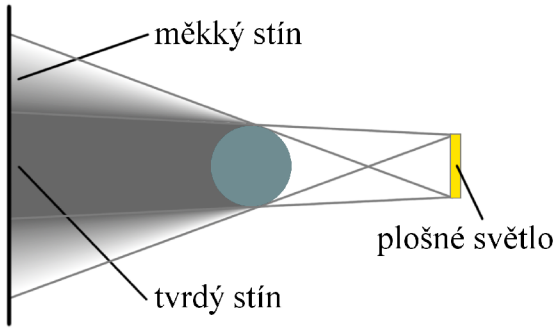
- nekonečná světla
- plošná světla
- bodová světla
- kuželovitá světla

Nekonečná světla jsou taková, která leží v nekonečné vzdálenosti a jejich světlo tedy dopadá na všechny objekty pod stejným úhlem. Jedná se o zjednodušení situace, kdy je světelný zdroj velmi daleko. Tohoto přístupu se například využívá, pokud chceme v počítačové grafice simulovat sluneční světlo. Jelikož počítáme s předpokladem, že je světlo nekonečně daleko, je celá scéna nasvětlena paprsky homogenně a všechny paprsky jsou rovnoběžné (viz 2.8). Pro definici takového světla pak stačí pouze směr paprsků (vektor) a jejich intenzita.



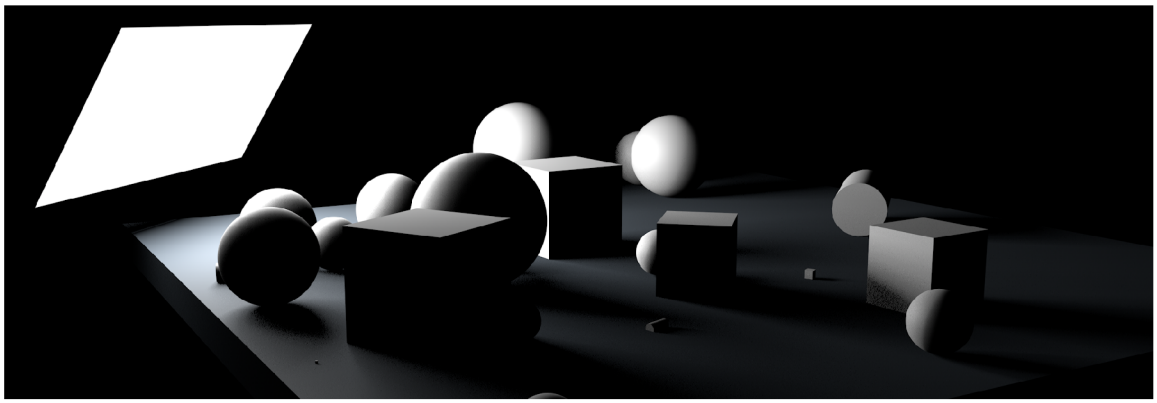
Obrázek 2.8: Ukázka nasvícení scény nekonečně vzdáleným zdrojem světla.

Použití plošných světel je fyzikálně nejpřesnějším přístupem. Jedná se o zdroje světla s určitou pozicí v prostoru a určitou plochou. Jejich použití je také nutné, pokud chceme dosáhnou fyzikálně přesných měkkých stínů. Množství světla dopadajícího na bod ve scéně nepřímo úměrně závisí na viditelné ploše světla z tohoto bodu viz 2.9.



Obrázek 2.9: Plošné světlo přirozeně tvoří měkké stíny.

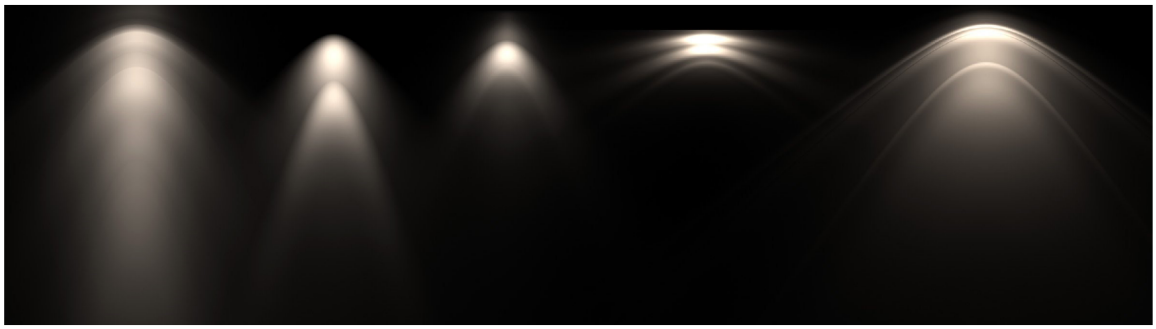
Tvar plošného světla může být různý. Často používaný tvar je obdélník (2.10), ale používají se další tvary, dvourozměrné (kruh, elipsa), či třírozměrné (krychle, válec, koule). Používají se dokonce i jednorozměrná (úsečka), ačkoli tato světla technicky nejsou plošná. Zvláštním typem plošných světel jsou objekty, které mají nadefinovanou svítivost. Přestože takové objekty jsou technicky plošná světla, často se s nimi počítá jako se součástí nepřímého osvětlení. Implementace nasvícení pomocí plošných světel je oproti ostatním typům světel poměrně náročná. Proto se, pokud je prioritou nižší náročnost, používají bodová světla.



Obrázek 2.10: Nasvícení scény pomocí plošného světla.

Bodová světla jsou nekonečně malé body v prostoru, které vyzařují světlo všemi směry. Jsou definovány svojí pozicí a intenzitou vyzařovaného světla. V některých implementacích se může lišit úbytek světla se vzdáleností. Často se počítá s nulovým úbytkem, přestože fyzikálně by měla intenzita ubývat s druhou mocninou vzdálenosti. Bodová světla se používají velmi často, jelikož je jejich základní implementace jednoduchá.

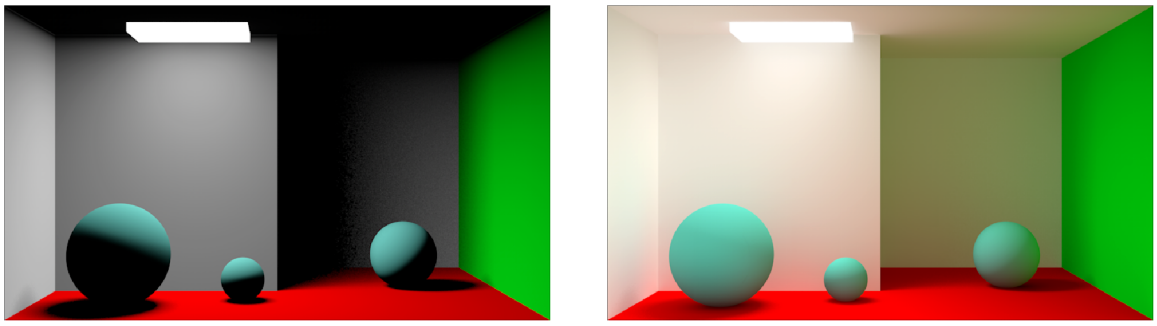
Kuželovitá světla jsou speciální typ bodových světel, které nevyzařují světlo všemi směry, ale pouze v určitém směru. Základní tvar vyzařovaného světla bývá jednoduchý kužel. Pro definici jiných tvarů se také používají IES soubory, přesně definující fotometrické vlastnosti daného svítidla, vycházející zpravidla z naměřených údajů reálných výrobků 2.11. Tyto soubory napomáhají věrnému zobrazení svítidel při vizualizaci, zpravidla je vydávají výrobci osvětlení.



Obrázek 2.11: Kuželovitá světla definovaná IES soubory.

2.3.2 Nepřímé osvětlení

Nepřímé osvětlení bývá označované jako globální iluminace (zkratka GI). Pokud se při nasvětlování scény počítá pouze s přímým osvětlením primárními světly, budou části modelů odvrácené od zdrojů světla (tedy ty, jejichž normálový vektor svírá se všemi vektory ke světelným zdrojům úhel větší nebo roven devadesáti stupňům) černé. Nepřímé osvětlení vychází z toho, že světlo vzešlé z primárního zdroje se odrazilo od objektu nebo se lomilo v průsvitném objektu, dále osvětluje zbytek scény. Výsledný efekt pak přidává na realismu, jak je ukázáno na obrázku 2.12.



Obrázek 2.12: Ukázka efektů nepřímého osvětlení.

Pro nepřímé osvětlení se využívá mnoho metod lišících se svým přístupem k simulaci šíření světla, přesností výsledků a výpočetní náročností. Mezi tyto metody spadá *ray tracing*, radiozita, mapování fotonů nebo například *voxel cone tracing*.

V případě *ray tracingu* se pro každý bod obrázku do scény *střílejí* paprsky, které poté určují, co na daném místě obrázku bude viditelné, a opětovným vysíláním paprsků ze *zasaženého* bodu se určí světelnost.

Výpočet radiozity pracuje na principu víceprůchodového počítání osvětlovacího modelu, kde se v dalších průchodech jako zdroje světla používají samotné objekty. Pro tuto metodu je třeba všechny objekty nejprve rozdělit na malé plošky.

Při použití metody mapování fotonů se ze zdroje světla náhodnými směry vysílají *fotony*, které poté dopadají na objekty a tím je osvětlují. Narozdíl od velké části ostatních metod berou fotony v potaz i průhledné či odrazivé materiály, tím vyváří světelné vzory – kaustiku (viz 2.13).

Metoda *voxel cone tracing* nejprve zvoxelizuje scénu, a pak počítá šíření světla pomocí kuželovitých paprsků. Výhoda této metody je, že ji možno využít na počítání osvětlení v reálném čase.

Obrázek 2.13: Kaustika – rozdíly v intenzitě dopadajícího světla způsobené refrakcí v průhledném materiálu.

Jako určitá náhrada či zjednodušení nepřímého osvětlení se v počítačové grafice využívají mapy prostředí.

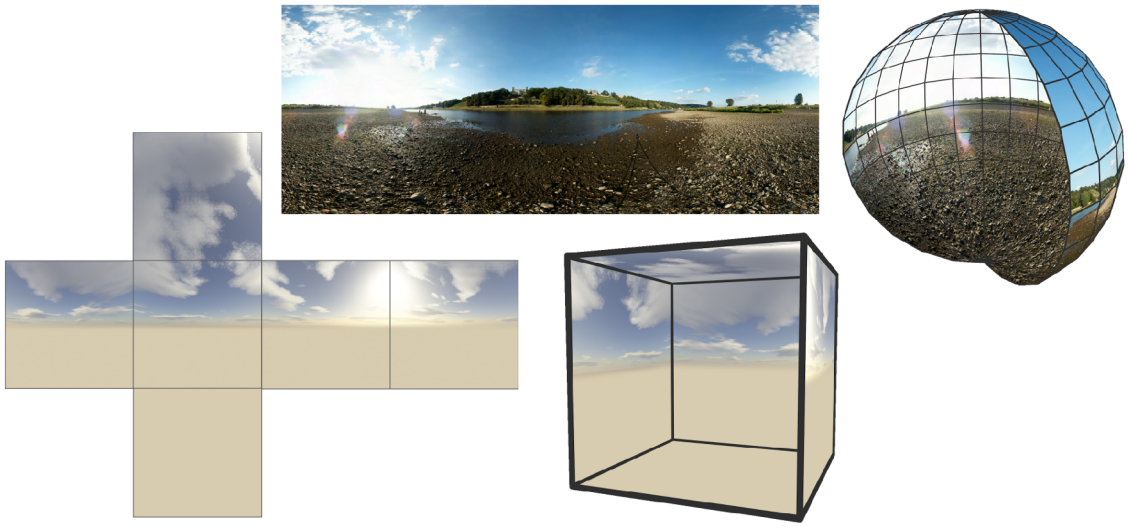
2.3.3 Mapa prostředí

Mapování prostředí (z originálu Evnvironment mapping, [3]), vychází z faktu, že velké objekty, které jsou velmi daleko od pozorovatele, při jeho pohybu působí, jako by stály na

místě; relativní posun je vůči jejich velikosti minimální. Je tedy možné jejich podobu a pozici zaznamenat do obrázku. Takový obrázek, neboli mapu prostředí, pak lze použít jako pozadí zobrazovaných objektů, a objekty samotné jej mohou i odrážet, jako na obrázku 2.14. Snížení výpočetních nároků v zobrazování obrázku namísto renderování všech viditelných objektů je značné. Pro potřeby počítačové grafiky je vhodné, aby takový obrázek pokryl veškeré okolí.

Obrázek 2.14: Ukázka mapování prostředí – mapa je zde vyfotografována a využívá se pro pozadí objektů a v odrazech.

Pro pokrytí pozadí scény je potřeba více snímků, případně jeden sférický. Sférický snímek se mapuje na kouli (obrázek 2.15). Zhotovení sférického obrázku je ovšem technicky poměrně náročné, a také zobrazení dokonalé koule je v počítačové grafice, zejména při používání polygonálních modelů, obtížné. Proto se často využívá více obrázků, které se namapují na stěny krychle, a k vykreslování prostředí se pak přistupuje stejně, jako při vykreslování jakéhokoliv jiného objektu [8]. Krychli s namapovanými snímky se říká *skybox*. Ukázka mapování obrázků na *skybox* je na obrázku 2.15.



Obrázek 2.15: Srovnání mapování sférického snímku na kouli a šesti různých snímků na krychli.

Při použití *skyboxu* pro pozadí se krychle renderuje jako nejvzdálenější objekt od pozorovatele. To znamená, že všechny ostatní objekty se zobrazují před ní. Pokud se *skybox* využívá pro přímou odrazivost nebo průhlednost, spočítá se vektor určující směr paprsku od *skyboxu*. Protože je *skybox* nekonečně daleko, všechny vektory se stejným směrem budou vracet totožnou barvu.

Kapitola 3

Návrh aplikace

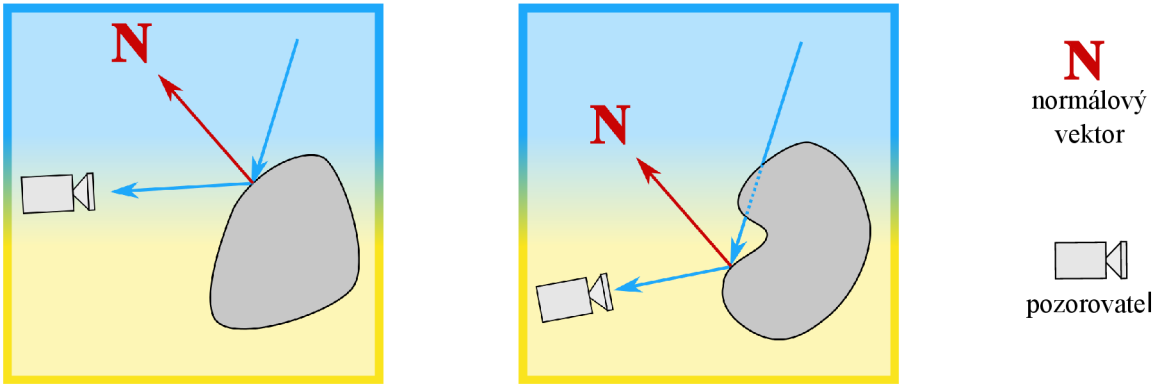
Aplikace má za úkol demonstrovat metodu Physically Base Shading. Rozhodl jsem se pro vytvoření aplikace ve stylu videohry, tedy běžící v reálném čase. Implementovat budu metody, běžící ve *vertex shaderu* nebo *fragment shaderu* OpenGL.

3.1 PBS Metody

Jedna z metod s velmi zřetelnými výsledky a fyzikální základem je přímá odrazivost. Správně implementovaná odrazivost musí odrážet světlo správným směrem, respektive ze správného směru pro pozorovatele, a se správnou intenzitou.

3.1.1 Odrazivost – směr

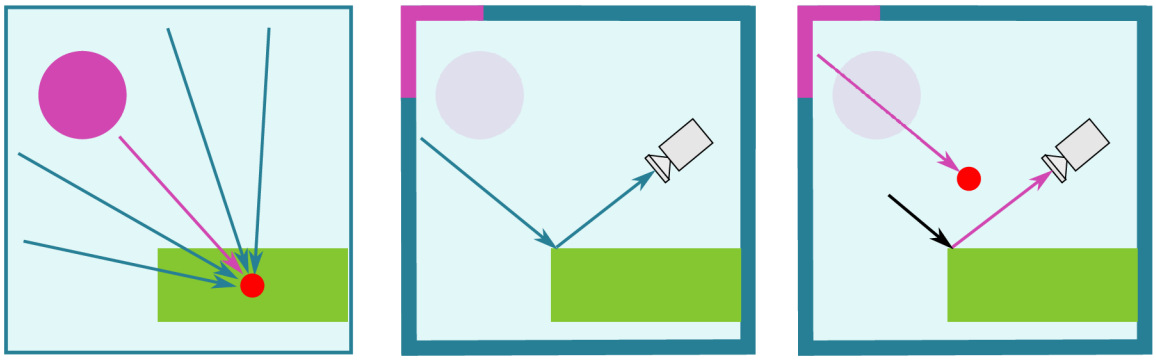
Správného směru odrazu se dosáhne trasovacími metodami, jako je například *raytracing*, či *pathtracing*. V některých případech je ale možné využít jiné metody pro dosáhnutí stejného, či podobného výsledku. Pokud vezmeme v úvahu scénu obsahující jediný objekt, který odráží nekonečně vzdálené pozadí, tj. *skybox*, budou všechny části povrchu se stejným normálový vektorem odrážet k pozorovateli stejné místo v pozadí. Bude tedy stačit z úhlu odrazu spočítat úhel dopadu paprsku. Jak ukazuje obrázek 3.1, toto platí pouze v případě, že povrch objektu neobsahuje žádné konvexní plochy, a nemůže tedy odrážet sám sebe. Pokud takové části obsahuje, může v některých místech docházet k nepřesnostem.



Obrázek 3.1: Pokud objekt obsahuje konvexní plochy (vpravo), může nastat situace, kdy je zjednodušený odraz chybný.

Tato metoda je ovšem použitelná pouze k odrážení pozadí. Pokud bude ve scéně více odrazivých objektů, nebudou se odrážet navzájem. Tento problém by bylo opět možné vyřešit použitím trasovacích metod, ovšem v případě, kdy se navzájem odráží dva či více objektů, by bylo třeba spočítat velmi mnoho odrazů pro dosažení realistického vzhledu.

Vzájemného odrážení objektů lze ale také dosáhnout úpravou předešlé metody, využívající *skybox*. Pokud před renderováním objektu pro něj vytvoříme *skybox*, ve kterém budou zachycené okolní objekty, budou tyto objekty viditelné i v následujícím odrazu. Toho se dosáhne tím, že se před samotným renderováním objektu vykreslí upravený skybox pro každý objekt. Skybox se ovšem takto renderuje pouze z jednoho bodu, čímž může vznikat chyba. Ačkoliv takový odraz bude zřejmě vypadat pěkně napohled, chyba bude tím větší, čím větší objekty a čím menší vzdálenost mezi nimi. To je způsobeno tím, že získání odrazu ze skyboxu je založeno na faktu, že je pozadí na skyboxu nekonečně daleko, což v případě objektů ve scéně neplatí. Chyba je ilustrována na obrázku 3.2.

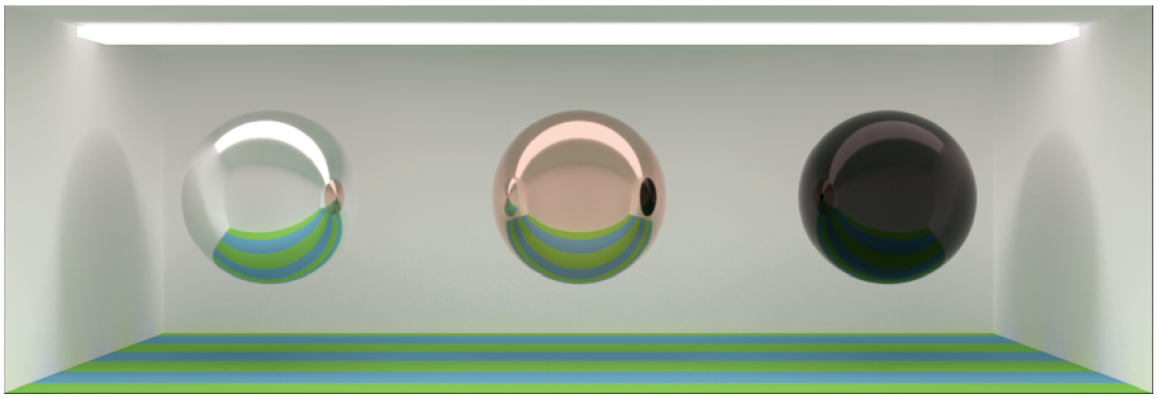


Obrázek 3.2: Chyba odrazu při *zapékání* okolí do *skyboxu* – pro odrážející objekt (zelený obdélník) se vypočítá *skybox* (vlevo). Při správně určeném odrazu, by z určité pozice pozorovatel viděl zelenomodré pozadí (uprostřed). Při použití *skyboxu* pro odraz se vypočítá vektor dopadu světla (černě) a aplikuje se na *skybox*, kvůli čemuž pozorovatel nesprávně vidí růžový objekt (vpravo).

Při renderování každé stěny skyboxu se stejným rozlišením, jako je výsledný obraz z aplikace, by došlo ke zvýšení náročnosti při jednom odrazivém objektu zhruba sedmkrát, proto se pro *skybox* často používá nižší rozlišení. Výpočetní náročnost takového řešení pak stoupá kvadraticky s počtem odrazivých objektů ve scéně, pro které je třeba generovat *skyboxy*. Přesto je při nižším počtu takových objektů ve scéně tato metoda relativně nenáročná.

3.1.2 Odrazivost – intenzita

Narozdíl od určování směru odraženého paprsku, se dá intenzita odrážení spočítat poměrně přesně i s relativně nízkou výpočetní náročností. Pro určení intenzity odrazu od určitého materiálu se u nekovových materiálů dá použít Schlickova aproximace Fresnelova koeficientu, v případě kovů pak její upravená verze, obě popsány zde 2.2.1. Pro nekovové materiály je tedy třeba zadat intenzitu odrazu pro tři vlnové délky světla odpovídající barvám modelu RGB. Pro kovy je nutno definovat koeficienty indexu lomu pro tyto vlnové délky, a to jak jejich reálné složky tak i složky imaginární.



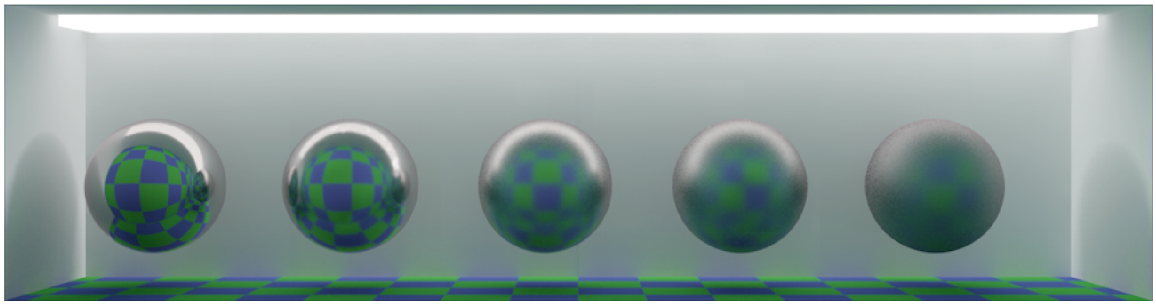
Obrázek 3.3: Rozdíl v intenzitách odrazu – maximální intenzita, kov (měď), nekov (plexisklo – PMMA).

Ve *fragment shaderu* se tedy vypočte barva odráženého místa z připraveného *skyboxu*. Pro každou barevnou složku této barvy (RGB) se spočítá koeficient odrazu pro odpovídající vlnovou délku světla a vynásobí se s touto barevnou složkou. Výsledná intenzita odrazu pak poměrně přesně odpovídá realitě (příklad 3.3).

Jediný přímý světelný zdroj v aplikaci bude nekonečné světlo (2.8). Jelikož je nekonečné světlo nekonečně malým bodem, při přímé odrazivosti bychom jeho odraz u dokonale hladkého materiálu viděli také nekonečně malý a z toho důvodu bude odraz světla zanedbán. Jiná situace ale nastává, pokud materiál nebude hladký.

3.1.3 Odrazivost – drsný povrch

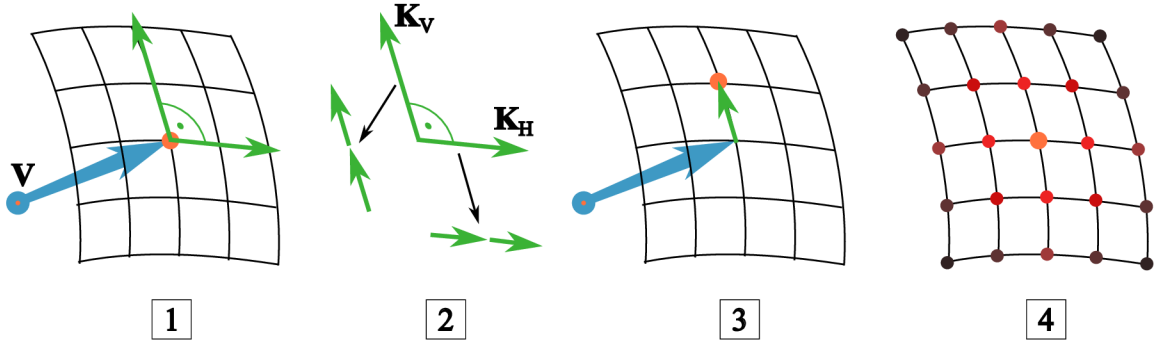
Na drsném povrchu materiálu se odrážené objekty zdají rozmazané. Každý bod na takovém povrchu odráží ne jeden konkrétní bod ve scéně, ale určitý výřez scény. S vyšší úrovní drsnosti velikost takového výřezu roste a odrážené objekty se zdají být více rozmazané. Je tedy nutné u materiálů tuto vlastnost definovat. Rozdíly různých hodnot drsnosti jsou ukázány na obrázku 3.4.



Obrázek 3.4: Různé hodnoty nastavení drsnosti od nejmenší (zleva) po nejvyšší (vpravo).

Pro výpočet takového odlesku je třeba modifikace metody, získávající hodnotu barvy ze *skyboxu*. Namísto chování, kdy funkce vrátí hodnotu barvy ze *skyboxu* podle vektoru (pro přehlednost dále označovaný jako $\mathbf{V}$), je třeba zprůměrovat více hodnot v takovém okolí bodu určeného $\mathbf{V}$, které odpovídá parametru drsnosti materiálu. Pro takové průměrování je vhodné použití například Gaussova rozmazání v dvouprůchodové variantě [6]. Koeficienty se pro daný poloměr dají předpočítat, čímž se ušetří výpočetní výkon. Na průměrování

hodnot je třeba získat matici těchto hodnot z okolí bodu určeného **V**. Protože přechody mezi stěnami krychle *skyboxu* jsou obtížné na výpočet, vypočítají se pouze vektory okolních bodů ze *skyboxu* a na získání jejich hodnot se použije zabudovaná funkce v *shaderu*. K výpočtu vektorů do matice se spočítají dva vektory (označené jako $\mathbf{K}_H$ a $\mathbf{K}_V$) kolmé k **V** tak, aby $\mathbf{K}_H$ byl rovnoběžný s rovinou **XZ** délka těchto vektorů značí poloměr rozmazání. Poté se k **V** přičítáním či odečítáním zlomků vektorů $\mathbf{K}_H$ a $\mathbf{K}_V$ (vektory se rozdělí podle počtu vzorků) získají souřadnice všech bodů matice. Postup je znázorněn na obrázku 3.5.

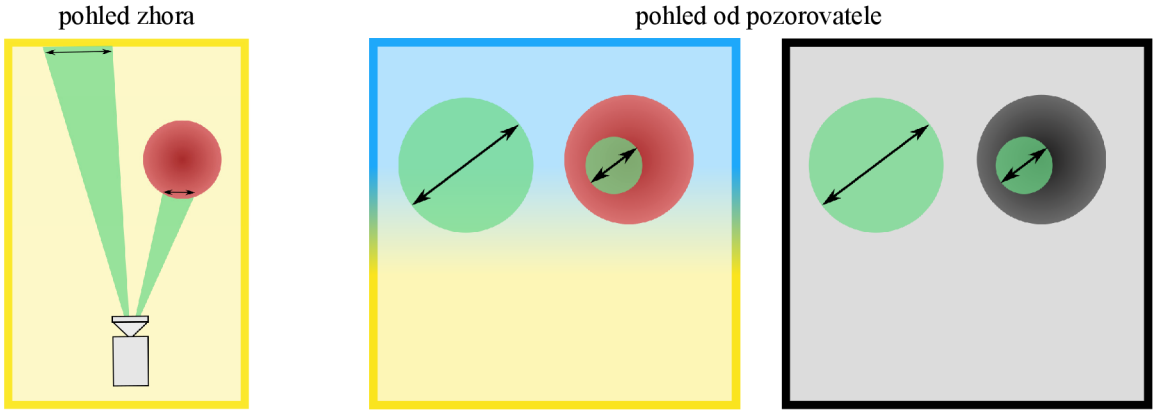


Obrázek 3.5: Získání matice bodů z vektoru. Nejprve se vypočítají kolmé vektory [1], ty se rozdělí na segmenty podle počtu bodů matice [2], postupně se přičítají k původnímu vektoru [3], pak se body dané těmito součty zprůměrují [4].

Pokud je odraz rozostřený, měl by v odraze být vidět i zdroj světla, přestože je nekonečně malý. Úhel, ve kterém se nekonečný zdroj světla odráží, závisí na drsnosti materiálu. Jelikož metoda pro získání barvy ze *skyboxu* s drsností počítá, je možné ji využít pro výpočet odrazu světelného zdroje. Jedna z možností je zopakovat výpočet, avšak namísto hodnot získaných ze *skyboxu* počítat přírůstky odrazu světelného zdroje. To by ovšem znatelně prodloužilo výpočet. Rychlejší by tedy bylo přírůstky odrazu světla počítat zároveň s vyhodnocováním hodnot *skyboxu*. Nejrychlejším způsobem by zřejmě bylo spočítat hodnotu pro daný fragment pouze jednou (ne pro každý přírůstek), a to pomocí rovnice Gaussovy funkce 3.1, kde x bude délka rozdílu mezi vektorem **V** a vektorem určujícím směr zdroje světla a σ bude vhodně zvolená podle délky vektoru $\mathbf{K}_H$ nebo $\mathbf{K}_V$.

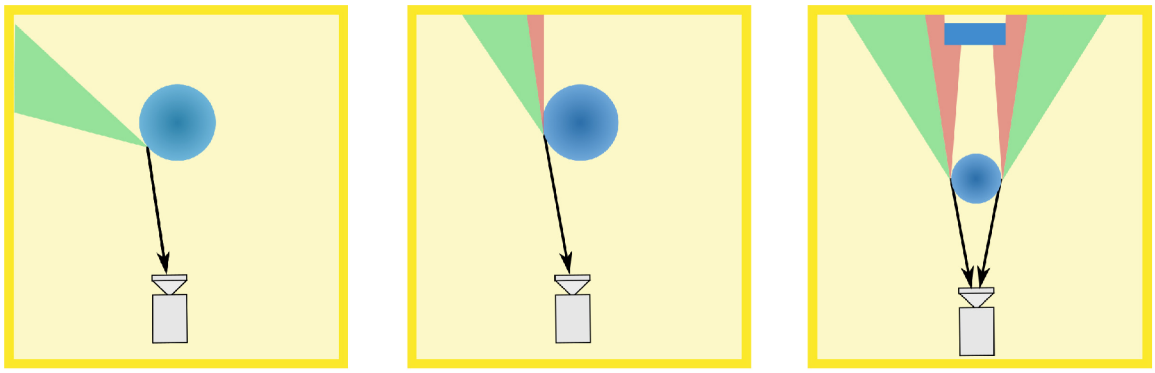
$$G = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{3.1}$$

Pokud má odrazivý objekt nastavenou drsnost – tedy odráží neostře. Odraz by měl být tím více rozostřený, čím větší je jeho vzdálenost od odrazivého objektu (obrázek 3.5 vlevo). Pokud by se odraz počítal některou z metod trasující paprsky, vznikl by takový efekt přirozeně. Pokud jsou ale všechny objekty *zapečené* do *skyboxu*, nezůstane o vzdálenosti předmětů informace a všechny objekty jsou v odrazu rozostřené tak, jako by ležely nekonečně daleko. Pokud se ovšem při renderování *skyboxu* používá *depth test*, lze data o vzdálenosti objektů uložit do textury. Při vytváření *skyboxu* je tedy možné zároveň vytvořit druhou texturu, která bude místo barevných hodnot uchovávat informaci o vzdálenosti. Pokud se bude používat metoda, při které se počítá rozostření pro každý bod zvlášť, je možné dosáhnout poměrně přesného rozostření v každém bodě tím, že se podle hodnoty z druhé textury upraví hodnoty vektorů udávající poloměr rozostření, jak je ukázáno na obrázku 3.6.



Obrázek 3.6: Při daném rozostření (vlevo, zde pozorovatel vidí rozostřeně) je na bližších objektech poloměr rozostření nižší než na vzdálenějších objektech. Pro dosáhnutí tohoto efektu (uprostřed) můžeme použít informace získaných s *depth textury* (vpravo).

Ačkoli takto upravená metoda může dosahovat v některých případech poměrně přesných výsledků, bude v některých místech vznikat chyba. Pokud se bude úhel mezi pozorovatelem a normálovým vektorem povrchu odrazivého materiálu blížit devadesáti stupňům, bude tato metoda počítat s body, od kterých by se světlo správně nemělo k pozorovateli dostat. Problém je ukázaný na obrázku 3.7. Je tedy možné, že na okrajích oválných objektů bude možné částečně vidět objekty, které by měly v daném pohledu zůstat skryty. Jelikož se ale tento problém projevuje s rozostřením odrazu, tak by jeho důsledky neměly být ve většině případů příliš zřetelné.



Obrázek 3.7: Chyba při rozostřování odrazu – vlevo je znázorněný případ, kdy rozostření vrací správný výsledek. Zelená výseč značí výřez scény, kterou pozorovatel vidí odraženou v daném bodě. Pokud se ovšem bude úhel mezi pozorovatelem a normálovým vektorem povrchu blížit pravému úhlu, bude pozorovatel v odraze vidět část scény, kterou by neměl. Tu značí červená výseč na obrázku uprostřed. Vpravo je pak demonstrována asi nejzřetelnější chyba, kdy pozorovatel v odraze vidí objekt, který by měl z dané pozice viditelný být neměl.

3.2 Zachování energie

Při odrazu světla od materiálu platí zákon zachování energie. Tedy světla vyzářeného odrazivostí a difúzí nemůže být více, než světla přijatého.

Pokud je materiál definován tak, že množství difúze a odrazivost nedávají dohromady více než sto procent, zachování energie bude dodrženo. V případech, kdy v definici materiálu bude v tomto chyba, je třeba tuto chybu ošetřit. Vhodným řešením bude v takových případech snížit hodnotu difúze. Tento problém může nastat pouze u materiálů, které mají zároveň odrazivost i difúzi, což jsou pouze lesklé nekovové materiály.

3.3 Aplikace

Aplikace demonstrující tyto metody by tedy měla umět načíst scénu, rozhodnout, které objekty se budou jak zobrazovat, a pak je zobrazit s tím, že pozorovatel bude moci procházet scénou v reálném čase.

3.3.1 Formát scény

Jako formát scény, který bude aplikace používat jsem zvolil Wavefront OBJ a MTL. Scéna v takovém formátu sestává se dvou souborů, soubor s příponou .obj obsahuje definici samotných objektů, ve volitelném souboru s příponou .mtl jsou definovány jejich materiály. Díky použití Assimpu bude aplikace schopná načíst spoustu různých formátů, OMB + MTL tedy volím z z toho důvodu, že je textový, tedy snadno modifikovatelný, a že umožňuje jednoduchou definici materiálů.

Protože jsem omezen parametry materiálu, které mohu do definice v .mtl zadat, a které AssimpModelLoader (4.1.3) načte a dokáže odlišit, je třeba vytvořit systém pro definici vlastností materiálu. Formát MTL má sice význam parametrů materiálů daný, ale tento systém není vhodný pro fyzikálně založenou definici materiálů, kterou tato aplikace požaduje. Význam parametrů materiálu je ukázaný v tabulce 3.1.

Tabulka 3.1: Význam vybraných parametrů v souboru MTL

	název parametru	**data**	**původní význam**
Konstanty:			
	Kd	3 × float	Diffuse color
	Ka	3 × float	Ambient color
	Ks	3 × float	Specular color
	Ns	1 × float	Shininess
Textury:			
	map_Kd	image	Diffuse texture map
	bump	image	bump texture map

Parametrů textur, které mohu v souboru MTL při využití modulu AssimpModelLoader použít, je více, v aplikaci jsou ale důležité zejména výše uvedené. Konstant, které je AssimpModelLoader schopen načíst a rozeznat je k dispozici deset. Konstanty jsou typu float. Pro určení typu materiálu použiji konstantu jedinou parametru *Ns*, původně určenou pro definici lesklosti. Parametr *Kd* využiji pro definici barvy, jak byla zamýšlena a parametr *Ka* pro definici odrazivosti. U kovových materiálů je parametr barvy nedůležitý, ale na definici odrazivosti je třeba šesti hodnot. Pokud tedy bude typem materiálu kov, pak s oběma parametry *Kd* i *Ka* bude zacházeno, jako s definicí odrazivosti. Parametr *Ks* využiji pro definici drsnosti materiálu. Parametr *map_Kd* použiji v původním významu, tedy pro texturu udávající barvu, a parametr *bump* pro normálovou texturu, jelikož formát MTL nemá parametr pro normálovou texturu. Shrnutí je v tabulce 3.2.

Tabulka 3.2: Sémantika vybraných MTL parametrů v aplikaci

Konstanty:

název parametru	**význam**
Ns	typ materiálu
Kd	barva, koeficient n u kovů
Ka	odrazivost, koeficient k u kovů
Ks	drsnost

Textury:

map_Kd	textura s barvou materiálu
bump	normálová textura

3.3.2 Rozdělení materiálů

Vzhledem k rozdílným metodám u různých materiálů se bude lišit způsob jejich zobrazování. Největší rozdíl v přístupu bude u kovových materiálů proti nekovovým. U nekovových materiálů zpravidla hraje hlavní roli v zobrazení difúze, a pak případná odrazivost. U kovů naopak odrazivost hlavní odrazivost, difúze se u nich neprojevuje. Proto bude vhodné materiály rozdělit a pro oba typy používat k zobrazování rozdílný *shader*. Jelikož je zobrazování odrazivosti výpočetně náročné, bude zřejmě vhodné u nekovových materiálů rozdělit zobrazování odrazivých a neodrazivých materiálů. Pokud nebude materiál odrazivý, nebude pro něj nutné vytvářet mapu okolí, čímž se výrazně ušetří výpočetní výkon.

Kapitola 4

Implementace

Součástí zadání je vytvoření aplikace demonstrující physically based shading. Metody, vhodné pro implementaci v OpenGL, které jsem se rozhodl implementovat, jsou probrané v kapitole návrh 3.

4.1 Použité nástroje

Práce je napsána v jazyce C++ verze 11. Pro zobrazení okna, vykreslovacího kontextu a odchytávání vstupů byla použita knihovna QT verze 5.5 a pro vykreslování obrazu na grafické kartě se využívá OpenGL verze 4.5. Další nástroje, použité v aplikaci jsou GPUEngine pro zjednodušení práce s OpenGL, Assimp pro načítání objektů a PicoPNG pro načítání textur.

4.1.1 GPUEngine

GPUEngine je knihovna určená pro pro práci s 3D scénou, její zpracování a renderování za pomocí grafické karty. Zjednodušuje práci s OpenGL díky zapouzdření funkcí do objektů. GPUEngine je vyvíjen v výzkumném centru informačních technologií na FIT VUT.

4.1.2 picoPNG

Na načítání textur je použit picoPNG, což je minimalistický dekodér obrázků ve formátu PNG. PicoPNG vychází z projektu LodePNG, kde v pozměněné formě slouží jako dekódovací jádro. Zajímavostí na picoPNG je, že se jedná pouze o jedinou funkci o zhruba pěti set řádcích v jazyce C++, v jediném souboru typu *.cpp*. Přesto dokáže tato funkce zpracovat libovolný standardní obrázek typu PNG, s různým barevným formátem či prokládáním. Obrázek vrací ve formátu standardního vektoru std :: vector. Velikost obrázku a jeho stran předává pomocí parametrů funkce. Pokud není parametrem zadáno jinak, funkce implicitně upraví formát na 32 bitový RGBA.

4.1.3 Assimp

Pro načtení 3D scény ze souboru 3D modelu se v této práci využívá knihovna Assimp. S funkcemi knihovny Assimp se pracuje za pomocí modulu AssimpModelLoader, vyvinutým Ing. Tomášem Starkou.

Jméno je zkratkou pro *Open Asset Import Library*. Assimp je multiplatformní knihovna s otevřeným zdrojovým kódem, určena k importování velkého množství různých 3D formátů 3D jednotným způsobem. Jelikož novější verze zvládá i export dat do 3D formátů, je použitelná i pro konvertování mezi různými typy souborů. Samotná knihovna je napsána v jazyce C++.

Modul AssimpModelLoader je napsán rovněž v jazyce C++ a skládá se ze dvou souborů typů *.cpp* a *.h*. Modul usnadňuje používání knihovny Assimp, zejména při využití knihovny GPUEngine. Načtení celé scény je realizováno zavoláním funkce loadScene() s jediným parametrem obsahující název souboru. Funkce pak vrátí ukazatel na objekt typu ge::sg::Scene, obsahující načtenou 3D scénu ze souboru, připravenou v přehledné struktuře, určenou pro snadnou práci v kombinaci s knihovnou GPUEngine.

4.2 Struktura aplikace

Základní strukturu aplikace tvoří modul mainwindow. V tomto modulu je definována třída OpenGLWindow, dědící ze dvou tříd, QWindow a QOpenGLFunctions. Třída definuje okno aplikace, kterému nastavuje kontext na vykreslování OpenGL. Dále pak definuje metody pro vykreslování a odchytávání událostí.

Okno výsledné aplikace je ovšem vytvořeno objektem MyWindow, která dědí třídu OpenGLWindow a implementuje konkrétní chování aplikace. Tato třída je definována v souboru main.cpp. Objekt této třídy obsahuje dva důležité objekty. Objekt třídy Model, obsahující scénu, a objekt kamery (ge::util::CameraObject).

4.2.1 Uložení scény

Scéna je po načtení uložena v objektu třídy Model, definovaném v modulu Models. Důvodem takového pojmenování je, že v aplikaci je celá scéna tvořena pouze jedním modelem. Protože ale model může obsahovat více *meshů*, není to problém. Model obsahuje vektor objektů třídy Mesh a pak skybox, který je také typu Mesh.

Objekt typu Mesh je definován také v modulu Models. Obsahuje ukazatel na VertexArrayObject, ukazatel na ProgramObject, textury, specifikace materiálu, skybox, skybox pro hloubkovou texturu středovou pozici *meshe*.

4.2.2 Odchytávání událostí

Odchytávání událostí, které zajišťuje vstupy z periferií, je řešeno v metodě event() třídy OpenGLWindow. Parametrem funkce je ukazatel na objekt QEvent, respektive objekt poděděný z QEvent. Konkrétní typ objektu se zjistí z atributu type. Typy událostí zpracovávané v aplikaci jsou:

- UpdateRequest
- Resize
- KeyPress
- KeyRelease
- MouseButtonPress
- MouseButtonRelease

- MouseMove
- Wheel

Podle typu události se ukazatel přetypuje na konkrétní typ a zavolá se metoda obsluhující konkrétní událost. Výjimkou je událost UpdateRequest, při které se rovnou volá vykreslovací funkce.

Metody obsluhující konkrétní typy událostí neprovádějí samy o sobě žádné akce, ale pouze nastavují hodnoty atributům. Další vyhodnocování událostí podle nastavených atributů provádí nezávisle na jejich nastavování.

Vyhodnocování je implementováno v metodě on_update třídy MyWindow. Tato metoda je volána vždy po vykreslení snímku. Pokud se změnila velikost okna, nastaví se příslušné nové rozměry pro OpenGL viewport a v objektu kamery. Pokud uživatel pohnul myší nebo držel klávesu pro pohyb, provede se určený posun kamery a atributy se vynulují.

4.2.3 Inicializace

Při spuštění programu se nejprve vytvoří okno aplikace. Po vytvoření okna se spustí inicializační metoda MyWindow::on_init. V té se nejdříve inicializuje OpenGL a poté se zavolá konstruktor třídy Model. Konstruktor přijímá jeden parametr, kterým je jméno souboru s modelem a invokuje metodu AssimpModelLoaderu pro načtení a zpracování scény. Ta vrátí ukazatel na objekt typu ge::sg::Scene. Pokud je hodnota ukazatele nulová, znamená to, že se soubor nepodařilo otevřít. V takovém případě se aplikace vypíše do konzole chybovou hlášku, a poté se ukončí. Pokud se scénu povedlo načíst, pokračuje se metodou createMeshes.

Metoda pro vytváření jednotlivých meshů spočítá, kolik jich scéna obsahuje a pak v cyklu volá jejich konstruktory. Poté zavolá konstruktor pro skybox.

Konstruktor meshe zjistí, v jakém pořadí jsou pro daný mesh uloženy pozice vrcholů, indexy polygonů a případné normálové vektory či UV koordináty. Ty poté použije na vytvoření VertexArrayObject. Z pozic všech vrcholů meshe se pomocí metody computeCenter spočítají minimální a maximální hodnoty ve všech osách, zprůměrováním minimálních a maximálních hodnot se vypočítá střed daného meshe. Konstruktor dále zavolá metodu pro načítání materiálů. Tato metoda projde všechny materiálové komponenty. Pokud se jedná o texturu, pokusí se jí pomocí PicoPNG načíst a uložit. Pokud se jedná a atribut materiálu, uloží si jeho hodnotu. Po načtení materiálu se rozhodne jeho typ a podle toho se vytvoří ProgramObject s příslušnými shadery. V případě že se nepodařilo načíst typ materiálu, je automaticky považován za nekovový matný. Konstruktor skyboxu vytvoří krychli a načte textury jednotlivých stran.

4.2.4 Postup vykreslování scény

Vykreslování probíhá invokací metody render, objektu třídy MyWindow, která spustí stejnojmennou metodu nad objektem třídy Model. Tato metoda přijímá v parametrech ukazatel na objekt kamery, aktuální rozlišení okna a pozici světla (respektive směr světla, jelikož se počítá s nekonečným zdrojem světla). Metoda je rozdělena na tři logické části – příprava, pomocné vykreslování a výsledné vykreslení obrazu.

V první části se spočítají matice, potřebné pro vykreslení scény. Jsou to matice model, View a Projection. View a Projection se získají z objektu kamery. Jejich součinem se získá celková matice MVP. Mimo to se z kamery získá pozice kamery a ViewRotation matice pro vykreslení skyboxu.

Druhá část metody slouží k vykreslení textur okolí pro jednotlivé meshe. Obsahuje cyklus, který pro meshe vytváří tyto textury – skyboxy. Pro každý mesh se nejprve ověří typ materiálu, pro matné materiály se skybox z důvodů optimalizace nevytváří. Pro vykreslování do textury se vytvoří FrameBuffer a velikost Viewportu se nastaví na rozměry jedné strany textury. Pokud mesh nemá vytvořené textury, vytvoří se. Textura udávající barvu používá formát GL_RGB, hloubková textura pak GL_DEPTH_COMPONENT32F. Také se vytvoří kamera, jejíž zorné pole se nastaví na devadesát stupňů. Pro každou stranu textury se framebufferu napojí daná strana pro obě textury. Vyčistí se barevný i hloubkový buffer a kameře se nastaví pozice, podle středu meshe, a správné natočení. Poté se vykreslí všechny objekty, kromě toho, pro který je konkrétní skybox určen.

Po vykreslení textur okolí do jednotlivých meshů se přenastaví viewport na předchozí hodnotu, tedy aktuální velikost okna. Nejprve se vykreslí skybox a poté všechny další meshe.

4.2.5 Vykreslení objektu

Vykreslení jediného objektu, tedy v tomto případě meshe, je realizováno metodou Mesh::render. Pro konkrétní mesh se nastaví správný ProgramObject. Do shaderu se vždy posílají matice (*Model*, *View*, *Projection*, *MVP*), směr zdroje světla, pozici kamery a velikost mapy okolí, i pokud není využita. Dále se posílá definice materiálu a informace o tom, které textury jsou pro daný objekt načteny.

Poté se do na texturovací jednotky navážou textury, pokud objekt nějaké má. Textury okolí se navážou vždy. Pokud nesou vytvořeny, naváže se místo nich textura skyboxu.

4.3 Shadery

Pro vykreslování na grafické kartě je v aplikaci vytvořeno 7 souborů s shadery v jazyce GLSL. Pro vykreslení skyboxu do pozadí určeny jsou dva shadery, typu vertex a fragment. Pro objekty je určen jeden společný fragment shader a pak tři vertex shadery, pro každý typ materiálu, které implementují samotné zobrazování.

4.3.1 Pomocné funkce

Shadery mají některé funkce stejné, případně podobné. Například funkce pro výpočet Fresnelova koeficientu podle Schlickovy aproximace má rozdílný výpočet u kovů (2.2) a nekovů (2.1), ale její použití je u obojího podobné.

Protože materiály mohou mít volitelné textury, jsou přítomny funkce, které podle parametrů určí, zda je textura načtená, a podle toho vrací danou hodnotu. Takové funkce jsou pro texturu s barvou, normálovou texturu a ambient oclusion texturu.

Důležitá je v shaderu funkce, která vrací hodnotu ze skyboxu, danou vektorem a poloměrem rozostření. Ta se používá pro výpočet odrazu. Pokud je poloměr rozostření roven nule, vrací pouze výsledek zabudované funkce texture. Pokud je však větší, postupuje podle metody, popsané v návrhu 3.1.3. Rozostření se počítá dvouprůchodovou Gaussovou metodou pro matici s pevnou velikostí 7×7 bodů, přičemž se mění pouze vzdálenost těchto bodů. Toto řešení jsem zvolil s důvodu, že při různé úrovní rozostření zachovává konstantní náročnost výpočtu. V opačném případě by při vyšší míře rozostření náročnost velmi rychle stoupala. Při konstantní velikosti matice lze navíc použít předpočítané konstanty pro průměrování, což také sníží výpočetní náročnost.

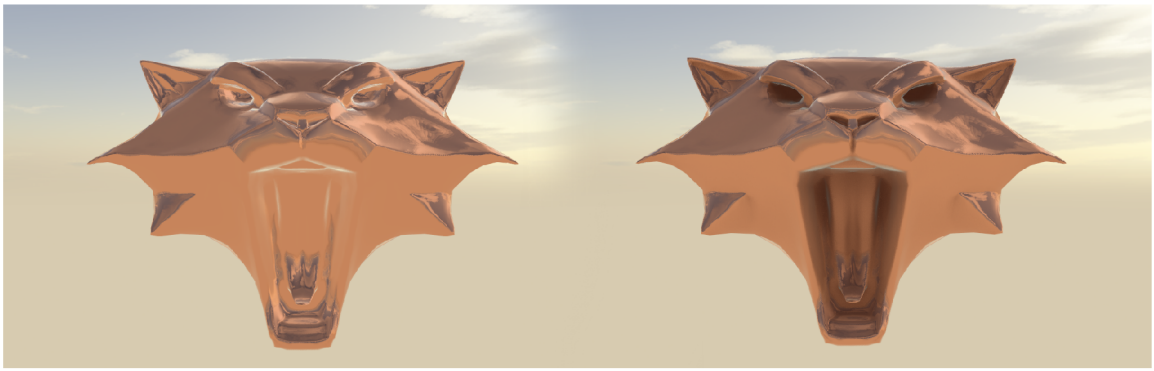
Z vektoru mířícím do textury se vypočítají kolmé vektory. Dále se přepočítají násobky kolmých vektorů, aby se nemusely zbytečně počítat vícekrát. Poté se v cyklu získají hodnoty pro všech 49 bodů, které se násobí koeficienty a sčítají do pole sedmi hodnot, připravených pro druhý průchod. V druhém průchodu se tyto hodnoty zprůměrují, čímž se získá konkrétní barva odrazu.

4.3.2 Vertex shader pro kovy

V případě kovů sestává postup pouze z určení správné barvy odrazu. Vypočtou se pomocné proměnné a zavolá se funkce pro získání odrazu. Jako parametr určující rozostřenost se předá hodnota z definice materiálu, vynásobená hodnotou získanou z hloubkové textury. Tím se dosáhne různého poloměru rozmazání u různě vzdálených objektů.

Pro barevné složky (RGB) se spočítá hodnota Fresnelova faktoru podle zadaných koeficientů n a k v definici materiálu, a jednotlivé barevné složky se jimi vynásobí.

Vzhledem k problému ukázaném na obrázku 3.1 se provede vizuální náprava, pokud je načtená ambient oclusion textura. Podle hodnoty získané z této mapy se ve tmavých místech, tedy v těch, kde jsou například otvory či záhyby, sníží odrazivost tím, že se jejich barva určí jako druhá mocnina jejich normální odrazivosti. Tím se v podstatě nasimuluje dvojitý odraz, ke kterým v takových místech dochází, efekt je předveden na obrázku 4.1.



Obrázek 4.1: Vizuální oprava problému s odrazem – za pomocí ambient oclusion textury je možné nasimulovat vícenásobný odraz v prohlubních objektu.

4.3.3 Vertex shader pro matné materiály

V případě vykreslování matného materiálu se zobrazí barva načtená z textury nebo zadaná atributem u materiálu. Ta se vynásobí cosinem úhlu, který svírá vůči světelnému zdroji. Aby strany objektu odvrácené od zdroje světla nebyly zcela černé, je přítomno ambientní nasvícení. Důvodem je, že v aplikaci není implementováno nepřímé osvětlení. Aby prohlubně v objektech vypadaly reálněji, tedy byly tmavší, je možné použít ambient oclusion texturu. Získaná barva se nakonec vynásobí hodnotou fresnelu, díky tomu se projeví odlesky při nízkých úhlech pohledu.

4.3.4 Vertex shader pro lesklé materiály

Lesklé materiály kombinují barevný kanál z matných materiálů a odrazivost (či lesk) kovových materiálů. V případě lesku se ale, narozdíl od kovů, používá na výpočet fresnelu funkce určená pro nekovové materiály. Než se obě složky sečtou, zkontroluje se, zda je

splněno zachování energie. Pokud je materiál nadefinován tak, že je zachování porušeno, spočítá se koeficient, kterým je nutno podělit barevnou složku na úkor odrazivosti. Tím se podělí všechny barevné složky, aby zůstal zachován odstín barvy a klesla pouze intenzita.

Kapitola 5

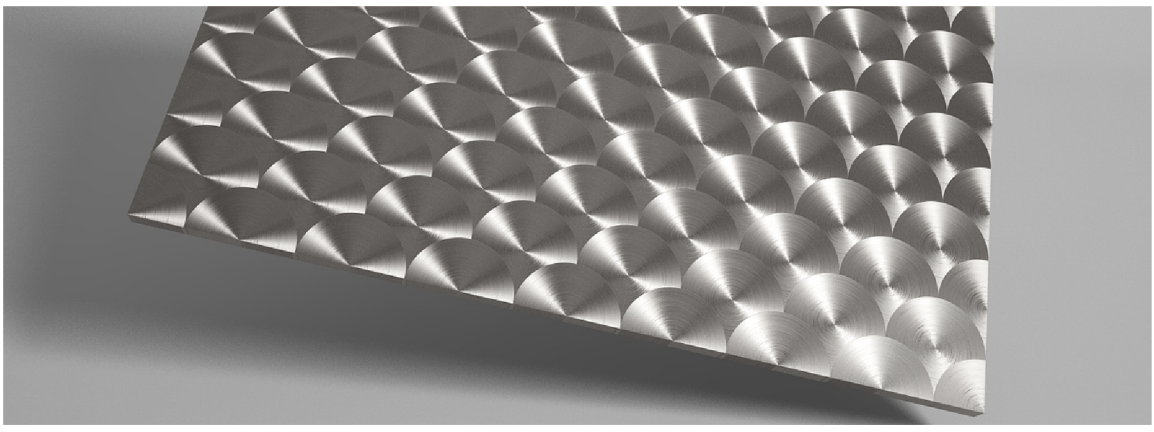
Možná budoucí rozšíření

V této práci je rozebrána implementace základního zobrazování kovových a nekovových materiálů. Metody zde popsané a implementované jsou ovšem schopny zobrazit pouze omezené množství reálných materiálů, kvůli nedostatků použitých metod mohou v zobrazování také vznikat chyby. Pokud se bude táto práce dále rozvíjet, například ve formě diplomové práce, je možné rozšířit její záběr a také opravit některé chyby, vznikajících díky aktuálně použitým metodám. Zde jsou uvedené některé příklady.

5.1 Opravy chyb v zobrazování odrazů

V kapitole návrh (3) jsou rozebrány metody pro výpočet směru a intenzity odrazivosti. U většiny metod jsou popsané možné chyby, které jejich použitím mohou vznikat. Je možné, že pozorovatel v odraze uvidí deformace, v některých odrazech dokonce celé objekty, které by viditelné být z daného pohledu neměly (3.7). Existují sofistikovanější algoritmy i korekce aktuálně použitých algoritmů, které tyto problémy řeší, a je možné je v rámci rozšíření implementovat pr přesnější zobrazování v odrazech.

5.2 Anisotropie



Obrázek 5.1: Anisotropie – rýhy v materiálu způsobené povrchovou úpravou dodávají povrchu pravidelný vzor.

Podstatná část této práce je věnována zobrazování kovů. Zabývá se mimo jiné taky neostrým odrazem, který je způsoben nerovným povrchem. Počítá však pouze se zcela rovnoměrným rozostřením. V reálném světě se často vyskytují v nerovnosti pravidelné vzory. Příklady takové nerovnosti jsou škrábance, které zcela mění vzhled povrchu, jak je ukázáno na obrázku 5.2. Tyto nerovnosti způsobují značné rozdíly v odrazivosti. Jednoduché pravidelné škrábance pouze způsobí v jedné ose mnohem větší rozostřenost než ve druhé, komplikovanější povrchové úpravy ale mohou vytvořit zajímavé vzory, jako například na obrázku 5.1.

Obrázek 5.2: Anisotropie – škrábance v povrchu materiálu výrazně ovlivňují jeho vzhled.

5.3 Průhledné a Průsvitné objekty

Typy materiálu, který v práci nejsou rozebírány ani implementovány, jsou průsvitné či průhledné materiály. Zobrazení průhlednosti je poměrně výpočetně náročné, často se pro výpočty používají trasovací metody. Při zobrazování je třeba počítat s vícenásobným lomem paprsku, navíc pokud povrch není hladký, pak při každém lomu vzniká rozostření (obrázek 5.3). U průhledných či průsvitných materiálů se často vyskytuje zároveň odrazivost.

Obrázek 5.3: Průhlednost, vlevo průhlednost s postupně stoupající drsností povrchu směrem vpravo.

5.4 Implementace nepřímého osvětlení

Nepřímé osvětlení bylo v části teorie sice rozebíráno (2.3.2), ale nebylo implementováno. Zůstává tedy jako námět možného budoucího rozšíření práce.

Kapitola 6

Závěr

V kapitole 2 byly popsány fyzikální principy šíření světla a jeho interakce s materiály. Byla zmíněna difúze i odrazivost a jevy, které je ovlivňují. U každého jevu bylo názorně ilustrováno, jaký má vliv na výslednou podobu materiálu. V případě Fresnelova jevu byly uvedeny aproximační rovnice pro jeho výpočet a rozdíl jeho chování u kovů a nekovů. Dále byly popsány různé způsoby osvětlení scény. Probrány byly přímé zdroje světla (světla bodová, plošná, kuželovitá a nekonečná) i některé metody nepřímého osvětlení scény a jejich výhody. Jako zjednodušení nepřímého osvětlení byly popsány mapy prostředí, které simulují okolí scény podle obrázku.

Byly probrány metody schopné pracovat v reálném čase. U těchto metod byly popsány jejich nedostatky a chyby při vykreslování, které mohou způsobovat, a navržena jejich možná implementace. Také byl navrhnut systém rozdělení materiálů podle typu a míry odrazivosti.

Na základě získaných znalostí byla vytvořena aplikace demonstrující vybrané metody. Věnuje se zejména zobrazení odrazivosti, s důrazem na věrné vykreslení kovů. Pro účely aplikace byl navrhnut systém fyzikální definice materiálů, založené na formátu Wavefront MTL. Jelikož tento formát není určen pro fyzikální definici, byl význam některých atributů pro účely demonstrace upraven. Aplikace je schopná zobrazit objekty kombinující kovové, nekovové lesklé a nekovové matné materiály. Aplikace má ovládání podobné videohrám z pohledu první osoby a podporuje ukládání aktuálních pohledů do souboru a také jejich načítání.

V kapitole 5 byla zmíněná možná budoucí rozšíření práce. Pokud se bude tato práce dále vyvíjet, je možné přidat podporu pro průhledné a průsvitné materiály, opravit nepřesnosti ve vykreslování a implementovat některé sofistikovanější osvětlovací modely. Dané metody by šly dále využít pro poměrně přesné zobrazení komplexních fyzikálně definovaných scén.

Literatura

[1] Alexander Keller, H. N., Stefan Heinrich: *Monte Carlo and Quasi-Monte Carlo Methods*. Springer Science and Business Media, 2007, ISBN 978-3-540-74496-2.

[2] Angel, E.: *Interactive Computer Graphics: A Top-Down Approach Using OpenGL*. Addison-Wesley, 2003, ISBN 978-0-321-31252-5.

[3] Greene, N.: *Environment Mapping and Other Applications of World Projections*. 1986, doi:10.1109/MCG.1986.276658.
URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4056759

[4] Hecht, E.: *Optics (2nd ed.)*. Addison-Wesley, 1987, ISBN 0-201-11609-X.

[5] István Lazányi, L. S.-K.: *Fresnel Term Approximations for Metals*. 2005.
URL https://otik.uk.zcu.cz/bitstream/handle/11025/11214/Lazanyi.pdf

[6] Mark Nixon, A. A.: *Feature Extraction and Image Processing*, Academic press. 2008.

[7] Schlick, C.: *An Inexpensive BRDF Model for Physically-based Rendering*. 1994, doi:10.1111/1467-8659.1330233.
URL http://www.cs.virginia.edu/~jdl/bib/appearance/analytic%20models/schlick94b.pdf

[8] W. Heidrich, H. S.: *View-Independent Environment Maps, Eurographics Workshop on Graphics Hardware*. 1998.

Přílohy

Seznam příloh

Příloha A

Ovládání aplikace

Aplikace obsahuje spustitelný soubor PhysicallyBasedShading.exe. Pro správnou funkčnost je třeba spouštět aplikaci na grafické kartě s podporou OpenGL 4.3 a vyšší.

Aplikace se ovládá za pomocí klávesnice a myši. Při držení levého tlačítka myši se otáčí kamera. Význam kláves je vypsán v tabulce.

Tabulka A.1: ovládání

Klávesy:		
	W	pohyb dopředu
	S	pohyb dozadu
	A	pohyb vlevo
	D	pohyb vpravo
	Space	pohyb nahoru
	Ctrl	pohyb dolů
	Esc	ukončit aplikaci
	Shift + [F1 - F4]	uložit pozici kamery
	[F1 - F4]	nahrát pozici kamery

Příloha B

Obsah CD

- Aplikace
- Zdrojové kódy aplikace
- Video