Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Zobrazovací metody založené na fyzikálních modelech

Bakalářská práce

Autor: Tomáš, Král
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Bruno Ježek, Ph.D.

Pracoviště: Katedra informatiky a kvantitativních metod

Hradec Králové Duben 2023

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 17.4.2023 Tomáš Král

Poděkování:

Děkuji vedoucímu bakalářské práce Ing. Bruno Ježkovi, Ph.D za metodické vedení práce, vstřícnost a odborné konzultace.

Anotace

V bakalářské práci byly prozkoumány zobrazovací metody založené na fyzikálních modelech – physically based rendering (PBR). V první části byly popsány teoretické aspekty PBR. Pozornost byla věnována zejména fyzikálně založeným dvousměrovým distribučním funkcím odrazu světla (BRDF) a modelům osvětlení založených na metodách PBR. V druhé části textu byl popsán postup implementace zvolených PBR metod pomocí programovatelných grafických karet. V rámci druhé části byla vyvinuta desktopová aplikace demonstrující zvolené zobrazovací metody.

Klíčová slova: PBR, BRDF, metody osvětlení, OpenGL

Annotation

Title: Physically based rendering methods

This bachelor thesis deals with physically based rendering methods (PBR). First, the theoretical aspects of PBR were described, mostly concerning physically-based bidirectional reflectance distribution functions (BRDF) and physically-based lighting models. The second part of the text describes the implementation of chosen PBR rendering methods using programmable GPUs. A desktop application demonstrating the chosen rendering methods was developed.

Keywords: PBR, BRDF, lighting methods, OpenGL

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Metody založené na fyzikálních modelech jsou zobrazovací metody, které využívají techniky, algoritmy a výpočty založené na zákonech popisujících chování světla při interakci s látkou. Anglicky se tyto metody nazývají „Physically based rendering“ neboli „PBR“. Tato práce je zaměřena především na „Physically based shading“, neboli „PBS“. Dalo by se říci, že PBS je podmnožina PBR, která se zaměřuje především na realistické zobrazování materiálů, kdežto PBR označuje fyzikálně založené techniky v rámci celého zobrazovacího řetězce, jako je například fyzikálně založená kamera či post-processing.

Přestože jsou PBR metody založené na fyzikálních modelech, nesnaží se o přesnou simulaci reality. Ta by vyžadovala simulaci světla jako elektromagnetického záření, což by samozřejmě bylo extrémně výpočetně náročné. Pro techniky PBR je typické využití různých aproximací či abstrakcí za toho předpokladu, že jejich využití povede k realisticky vypadajícím výsledkům.

Metody PBS se dají chápat jako protiklad empirických zobrazovacích technik, jako je například Phongovo stínování. V real-time grafice se techniky PBS liší od empirických technik několika aspekty. Za prvé využívají fyzikálně smysluplné parametry, jako například drsnost povrchu a za druhé využívají matematicky přesnější modely odrazu světla od povrchu. Využití technik PBS také umožňuje snadnější vytváření 3D modelů a jejich převod mezi různými 3D aplikacemi [1].

Cílem práce je prozkoumat přístupy pro zobrazení scény s využitím fyzikálně založených zobrazovacích metod – physically based rendering (PBR). Pro vhodnou scénu navrhnout, implementovat a otestovat řešení pracující v reálném čase.

V první části práce jsou popsány teoretické aspekty PBR. Pozornost je věnována zejména dvousměrovým distribučním funkcím odrazu světla. Jejich vlastnosti jsou popsány v kapitole 3. V kapitole 4 je popsán fyzikálně založený způsob modelování těchto funkcí. Modelům osvětlení je věnována kapitola 5. V kapitole 6 je vysvětlen způsob použití GLTF modelů s PBR metodami.

Druhá část práce se zabývá návrhem a implementací aplikace používající zobrazovací metody PBR. V implementační části jsou popsány použité technologie a

je nastíněna struktura softwarového řešení. Pozornost je věnována především implementaci osvětlovacího modelu.

2 Sledované fyzikální jevy

Při zobrazování 3D scény metodami počítačové grafiky, a to zejména při renderování v reálném čase, není z výkonových i algoritmických důvodů možné simulovat všechny fyzikální vlastnosti světla, a proto je nutné světlo abstrahovat. Z pohledu fyziky je světlo viditelné spektrum elektromagnetického záření, které se prostředím šíří jako vlna. V real-time grafice je běžné z velké části ignorovat jevy vycházející z vlnové podstaty světla jako například polarizace nebo difrakce. Na světlo se tedy nahlíží z pohledu geometrické optiky, tzn. počítá se s představou světelného paprsku.

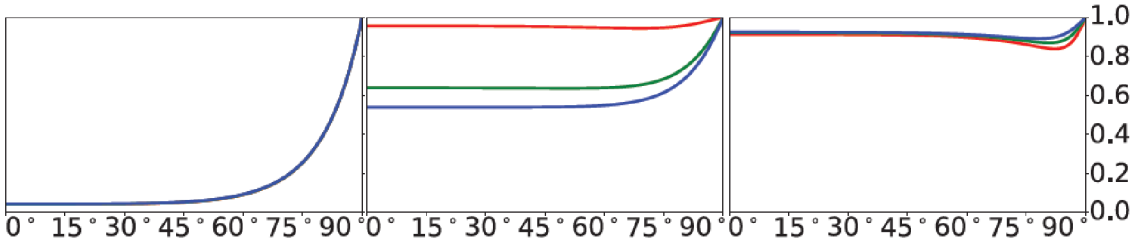
Další abstrakcí od fyzikální podstaty světla je reprezentace barvy. Grafické aplikace běžně renderují pomocí barevného modelu RGB, kdežto z pohledu fyziky je barva určena distribucí energie ve viditelném spektru[1]. Renderování pomocí barevných složek RGB může ve specifických situacích produkovat špatné výsledky, ale pro typické aplikace je tento způsob dostačující. Spektrální renderování je využíváno ve filmovém průmyslu například firmou Weta FX [2] či ve výzkumných projektech [3].

2.1 Odraz světla

Odraz světla od povrchu se řídí několika zákony. Z pohledu fyziky je povrch objektu rozhraní mezi 2 prostředími, které mají odlišný index lomu. V této práci budu předpokládat, že „vnější" prostředí je vzduch, jehož index lomu je přibližně 1. Při kontaktu světla s rozhraním se část světla odrazí a druhá část lomí. Poměr odraženého a lomeného světla popisují tzv. Fresnelovy rovnice. Ty jsou poměrně komplikované a jejich přímé využití není v real-time grafice vhodné, proto zde nebudou uvedeny. Pro zjednodušení bude pro označení Fresnelových rovnic použita funkce $F(\theta)$, kde θ je úhel mezi normálou rozhraní a světelným paprskem na něj dopadajícím. $F(\theta)$ je zároveň závislá na vlnové délce světla, což v real-time grafice znamená rozdělení výpočtu pro jednotlivé RGB složky. Příklady hodnot $F(\theta)$ pro 3 různé materiály jsou vidět na Obr. 1. $F(\theta)$ má charakteristickou vlastnost: poměr

[1] Anglicky „Spectral Power Distribution".

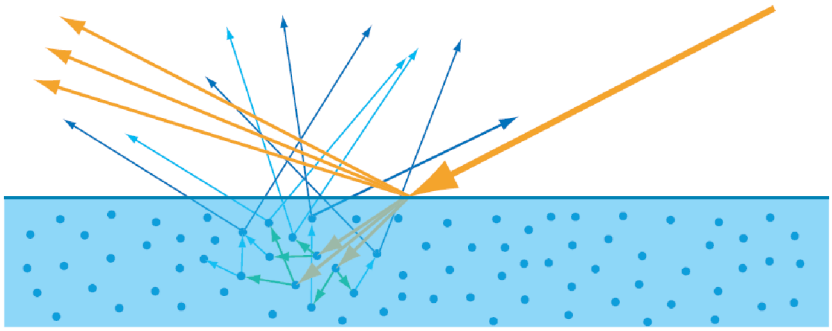
odraženého světla se pro úhly θ blízké 90° blíží jedné. Tato vlastnost platí pro všechny materiály a označuje se jako „Fresnelův efekt“.



Obr. 1 Příklady hodnot Fresnelovy funkce (zleva): sklo, měď, hliník. Převzato z Real-Time Rendering [4].

2.2 Podpovrchový rozptyl

Chování lomeného světla závisí na vlastnostech prostředí, ve kterém se lomené světlo pohybuje. Například kovy téměř všechno lomené světlo absorbují. Naopak v nevodičích dohází k podpovrchovému rozptylu. Podpovrchový rozptyl označuje proces, kdy je lomené světlo v podstatě náhodně rozptýleno uvnitř objektu. To je následně buďto absorbováno nebo opět vyzářeno v téměř náhodných směrech, viz Obr. 2.



Obr. 2 Podpovrchový rozptyl. Převzato z Real-Time Rendering [4].

Pokud je vizuální vzdálenost mezi místem dopadu světla a místem opětovného vyzáření malá, jedná se o lokální podpovrchový rozptyl, nazývaný také „difusní složka“. Přímo odražená část světla se nazývá zrcadlová resp. „spekulární složka“. Obě složky jsou součásti modelu odraženého světla.

V případě, že je daná vzdálenost vysoká, hovoří se o globálním podpovrchovém rozptylu [4]. To je poměrně významný jev pro specifické materiály jako například listy rostlin, lidská kůže nebo další průsvitné materiály. V real-time

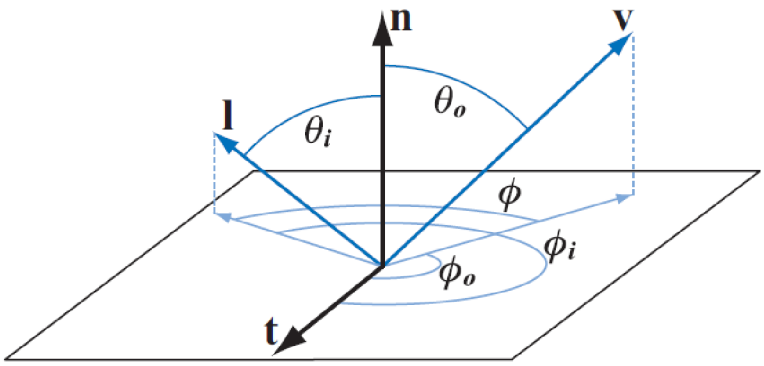
grafice je globální podpovrchový rozptyl považován za speciální případ a jeho renderování využívá specifické techniky [5, 6].

3 BRDF a Rovnice odrazu

3.1 BRDF

Dvousměrová distribuční funkce odrazu světla, anglicky „bidirectional reflectance distribution function“ (BRDF) hraje zásadní roli v oblasti PBR, protože precizně popisuje odraz světla na zobrazovaném objektu. Pro další typy materiálů jako jsou například částečně průhledné materiály je třeba BRDF dále rozšířit. Příkladem může být funkce BTD, „bidirectional transmittance distribution function“. Konkrétní BRDF lze chápat jako synonymum slova „materiál“.

BRDF má tvar $f_{brdf}(\mathbf{p}, \mathbf{v}, \mathbf{l})$ a označuje poměr odraženého světla ve směru $\mathbf{v}$ ku příchozímu světlu ze směru $\mathbf{l}$ v bodě $\mathbf{p}$. Přesněji řečeno, pro určitý směr $\mathbf{l}$ určuje BRDF distribuci odraženého světla do všech možných směrů $\mathbf{v}$ v hemisféře orientované okolo normály $\mathbf{n}$. Místo vektorů $\mathbf{l}$ a $\mathbf{v}$ lze jako parametry BRDF použít úhly ϕ_i, ϕ_o, θ_i a θ_o.



Obr. 3 Ilustrace BRDF.
Převzato z Real-Time Rendering [4].

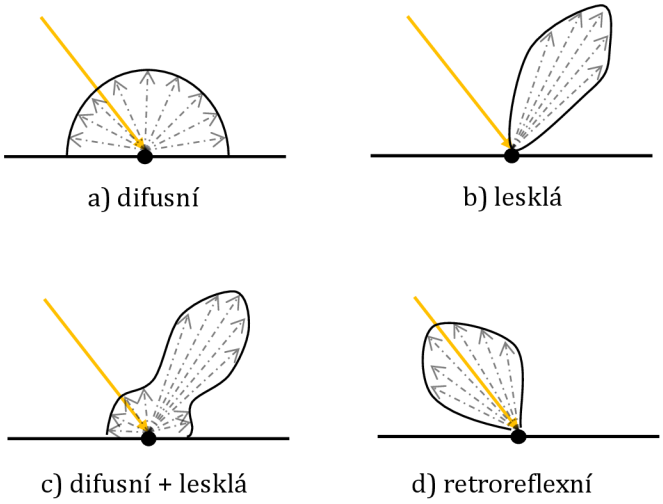
Aby byla BRDF fyzikálně smysluplná, musí splňovat několik vlastností [7]:

- Pozitivita: $f_{brdf}(\mathbf{p}, \mathbf{v}, \mathbf{l}) \geq 0$
- Symetrie: $f_{brdf}(\mathbf{p}, \mathbf{v}, \mathbf{l}) = f_{brdf}(\mathbf{p}, \mathbf{l}, \mathbf{v})$
- Zachování energie: $\int_{\Omega} f_{brdf}(\mathbf{p}, \mathbf{v}, \mathbf{l})(\mathbf{l} \cdot \mathbf{n})\mathrm{d}\mathbf{l} \leq 1$

Symetrie znamená, že se výsledek BRDF nezmění, pokud se prohodí vektory **v** a **l**. Vlastnost zachování energie zaručuje, že množství odraženého světla nebude vyšší než množství příchozího světla. BRDF je pozitivní – to znamená, že její hodnoty mohou dosahovat nekonečna. Hodnotu BRDF pro konkrétní dvojici vektorů **v** a **l** proto nelze interpretovat jako poměr odraženého světla, ale pouze jako hustotu pravděpodobnosti.

BRDF se dají rozdělit na izotropní či anizotropní. Izotropní BRDF jsou invariantní vůči rotaci podle osy **n** (záleží pouze na vzájemném úhlu ϕ). U anizotropních BRDF záleží také na jednotlivých úhlech ϕ_i, ϕ_o vzhledem k tangentě, viz Obr. 3. Typickým anizotropním materiálem je například broušený kov nebo přírodní vlákna.

Někdy se lze setkat s notací $f_{brdf}(\mathbf{p}, \mathbf{v}, \mathbf{l}, \lambda)$, kde λ označuje parametr vlnové délky světla, nicméně v této práci budu pracovat pouze s RGB hodnotami, jak je v real-time grafice zvykem. Na Obr. 4 je zobrazeno několik typických BRDF.



Obr. 4 Ilustrace typických BRDF.
Vlastní tvorba.

3.1.1 Naměřené BRDF

Jedním ze způsobů vyjádření BRDF je zadání pomocí tabulky naměřených hodnot. V takovém případě je v laboratoři změřen vzorek materiálu a jeho BRDF hodnoty jsou zapsány do tabulky. Tato reprezentace je velmi přesná a lze ji jednoduše využít v rendereru. Nevýhodou je zpravidla vysoká paměťová náročnost.

Velmi známá sada naměřených BRDF je tzv. MERL BRDF databáze [8]. Ta obsahuje data více než 100 izotropních materiálů. Podle Sébastiena Lagarda jsou tato data nepřesná pro úhly téměř kolmé na normálu. Nicméně mohou být užitečná pro porovnání analytických BRDF modelů s realitou [9].

Další databáze, která stojí za zmínku pochází z Ústavu teorie informace a automatizace AV ČR – tzv. UTIA BRDF database [10]. Autoři se naopak soustředili na měření anizotropních BRDF.

3.1.2 Analytické BRDF modely

Jak již bylo zmíněno, použití naměřených BRDF při renderování je sice přesné, ale trpí vysokou paměťovou náročností – například jeden materiál z MERL databáze obsahuje přibližně 30 MB dat, což je pro náročné grafické aplikace nepřijatelné. Primárním úkolem PBR je nalézt vhodné analytické BRDF modely - tj. BRDF, jež jsou popsány matematickou rovnicí, kterou lze jednoduše vyhodnotit. Tyto modely jsou typicky parametrizovány materiálovými vlastnostmi, jako je například drsnost povrchu, barva a podobně.

Analytických modelů BRDF existuje mnoho [11]. Pro účely této práce je vhodné je rozdělit na empirické, mezi které patří např. Blinn-Phong, Ward, Lafortune a na fyzikálně založené, do kterých řadíme např. Cook-Torrance, Oren-Nayar, Ashikhmin-Shirley nebo He et al.

Při výzkumu fyzikálně založených BRDF je žádoucí porovnávat vytvořené modely s realitou. Tím se zabýval například tým z MIT CSAIL [12]. Autoři využili již zmíněnou MERL databázi a snažili se „napasovat“ parametry některých analytických BRDF modelů na naměřená data. Podle autorů podávají fyzikálně založené BRDF (Cook-Torrance a Ashikhmin-Shirley) lepší výsledky než ostatní (Blinn-Phong, Ward a Lafortune).

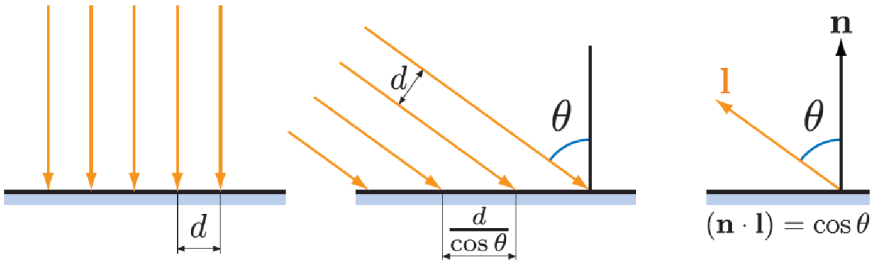
BRDF modely typicky počítají spekulární a difusní složky separátně, což umožňuje používat jeden model na širokou škálu materiálů. Takové modely mají tvar $f_{brdf} = df_{diffuse} + sf_{specular}$. Spekulárním modelům bude věnována kapitola 4.1 a difusním kapitola 4.2.

3.2 Rovnice odrazu

Rovnice odrazu je fundamentální rovnice počítačové grafiky, protože matematicky popisuje výpočet osvětlení pro libovolný bod v prostoru. Vychází ze známé zobrazovací rovnice a má přibližně následující tvar [7]:

$$L_{out}(\mathbf{p},\mathbf{v}) = \int_{\Omega} f_{brdf}(\mathbf{p},\mathbf{v},\mathbf{l_{in}}) L_{in}(\mathbf{p},\mathbf{l_{in}}) |\cos\theta_i| d\mathbf{l_{in}} \qquad \textbf{(1)}$$

Popisuje, jaké množství světla bude odcházet do směru **v** při integraci veškerého osvětlení L_{in} z jednotkové hemisféry Ω orientované podle normály **n**. Množství odchozího světla záleží na BRDF, na množství osvětlení $L_{in}(\mathbf{p},\mathbf{l_{in}})$ a na Lambertově zákonu. Lambertův zákon popisuje hustotu světelných paprsků dopadajících na plochu, která je přímo úměrná úhlu θ_i, viz Obr. 5.



Obr. 5 Lambertův zákon.
Převzato z Real-Time Rendering [4].

3.2.1 Monte-Carlo integrace a vzorkování podle důležitosti

Integrál z rovnice 1 nelze jednoduše vyhodnotit – nemá obecné analytické řešení. Proto se pro jeho odhad využívají numerické metody. Metodou, která se často používá v počítačové grafice je Monte-Carlo integrace [7].

Monte-Carlo umožňuje odhad libovolného integrálu ∫ f(x) pomocí vyhodnocení dostatečně velkého množství vzorků f(x) v náhodných místech definičního oboru funkce f, viz rovnice 2. V nejjednodušší variantě Monte-Carlo má náhodná proměnná X_i rovnoměrné rozdělení.

$$\int f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} \qquad \textbf{(2)}$$

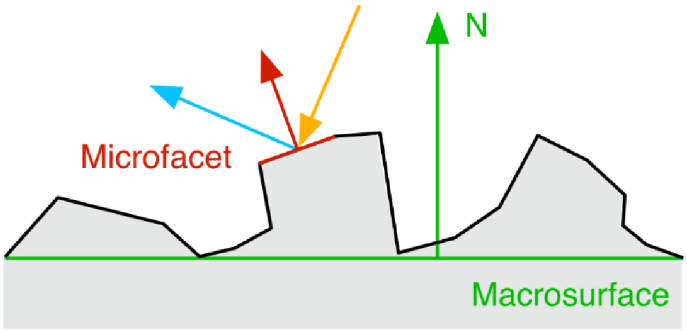
Nevýhodou metody Monte-Carlo je, že je potřeba vyhodnotit poměrně vysoké množství vzorků funkce f(x). Pro snížení počtu vzorků nutných k dosažení dobrého

odhadu integrálu lze použít například metodu vzorkování podle důležitosti (importance sampling). Vzorkování podle důležitosti vychází z toho, že pokud je rozdělení pravděpodobnosti $p(X_i)$ podobné funkci $f(X_i)$, pak bude odhad integrálu rychleji konvergovat ke správnému výsledku [7].

4 BRDF modely

4.1 BRDF modely založené na mikroplošk ách

V dnešní době jsou nejpoužívanější fyzikálně založené analytické BRDF modely založeny na teorii mikroplošek, kterou do počítačové grafiky zavedli Cook a Torrance [13]. Mikroplošky jsou ploché útvary, které tvoří mikroskopickou geometrii povrchu. To znamená, že jeden pixel může reprezentovat vysoké množství mikroplošek. Ty nejsou renderovány samostatně, ale jejich vliv na odraz světla je spočítán statisticky.



Obr. 6 Vizualizace mikroplošek.
Převzato z Physically Based Rendering in Filament [14].

Způsob odrazu světla od jednotlivých mikroplošek určuje mikroplošková BRDF. V této práci byl kladen důraz na spekulární modely, kde jednotlivé mikroplošky odráží světlo zrcadlově. Existují však BRDF modely jako například Oren-Nayar [15], které vychází z difusní mikroploškové BRDF.

Při použití zrcadlové mikroploškové BRDF má spekulární model následující tvar [7]:

$$f_{specular} = \frac{D(\mathbf{h})F(\mathbf{v},\mathbf{h})G(\mathbf{l},\mathbf{v},\mathbf{h})}{4\cos\theta_o\cos\theta_i} \qquad (3)$$

V této funkci vektor **h** představuje tzv. „halfway“ vektor, tedy vektor, který v polovině protíná úhel mezi vektory **l** a **v**. Funkce $D(\mathbf{h})$ je distribuční funkce normál, $F(\mathbf{v}, \mathbf{h})$ je Fresnelova funkce a $G(\mathbf{l}, \mathbf{v}, \mathbf{h})$ označuje funkci zastínění a maskování. Výhodou tohoto modelu je, že tyto 3 funkce nemusí mít konkrétní tvar. Je ale důležité, aby měly správné vlastnosti. Proto je tento model občas označován jako „obecný spekulární mikroploškový BRDF model“. Někdy je také označován jako Cook-Torrance model. Původní formulace Cook-Torrance modelu však obsahovala jiné $D(\mathbf{h})$, $F(\mathbf{v}, \mathbf{h})$, $G(\mathbf{l}, \mathbf{v}, \mathbf{h})$ funkce než ty, které se používají v dnešní době.

Některé publikace [14] vyjadřují spekulární model pomocí tzv. „visibility“ funkce: $V(\mathbf{l}, \mathbf{v}, \mathbf{h}) = \frac{G(\mathbf{l},\mathbf{v},\mathbf{h})}{4\cos\theta_o \cos\theta_i}$. Výsledná spekulární BRDF má poté tvar: $f_{specular} = D(\mathbf{h})F(\mathbf{v}, \mathbf{h})V(\mathbf{l}, \mathbf{v}, \mathbf{h})$.

4.1.1 Distribuční funkce normál

Distribuční funkce normál (DFN) určuje statistické rozdělení normál mikroplošek. Ze všech funkcí používaných v tomto spekulárním modelu má největší význam pro vzhled a přesnost výsledného modelu. Jednotlivé mikroplošky odrážejí světlo perfektně zrcadlově. Proto se na odrazu světla do určitého úhlu pohledu podílejí jen ty mikroplošky, jejichž normála je rovna vektoru **h**. Například DFN matného materiálu by měla poměrně rovnoměrné rozdělení, kdežto DFN zrcadlového materiálu by byla téměř všude nulová, pouze pro vektory **h** blízké normále by dosahovala maximálních hodnot. DFN jsou často definovány ještě pomocí parametru α, který určuje drsnost povrchu. Tím je umožněno jedné DFN modelovat širokou škálu materiálů. Hodnota parametru α se většinou pohybuje od 0 (jemný povrch) do 1 (hrubý povrch).

Aby byla DFN fyzikálně smysluplná, musí být normalizována. To znamená, že musí platit rovnice 4 [4]. Proměnná **m** označuje normály mikroplošek z jednotkové sféry Θ. Vektor **n** je normála makropovrchu.

$$\int_{\mathbf{m}\in\Theta} D(\mathbf{m})(\mathbf{n}\cdot\mathbf{m})d\mathbf{m} = 1 \qquad \textbf{(4)}$$

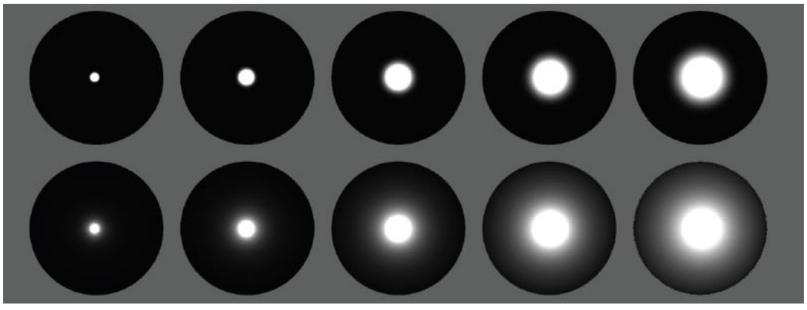
V dnešní době je nejpoužívanější DFN tzv. Trowbridge-Reitz [9, 16–19], častěji známá pod jménem „GGX“ [20]. Kromě GGX existují i další DFN, jako třeba

Beckmann DFN, ale ty zde pro stručnost nebudou uvedeny, protože GGX lze považovat za de-facto standard. GGX distribuce má oproti jiným známým distribucím určité výhody. Podle Sébastiena Lagarda více odpovídá datům z MERL databáze než ostatní distribuce [9]. Zároveň je oblíbená kvůli své vizuální kvalitě – oproti ostatním DFN má plynulejší přechod u okrajů spekulárního odlesku, viz Obr. 7. Tvar izotropní GGX je uveden v rovnici 5 [20]. V původním znění figurují funkce cosinus a tangens, které jsou v této rovnici nahrazeny trigonometricky identickými výrazy, které jsou jednoduší pro výpočet na GPU.

$$\mathrm{D}(\mathbf{m}) = \frac{\mathrm{H}^+(\mathbf{n}\cdot\mathbf{m})\alpha^2}{\pi\left(1+(\mathbf{n}\cdot\mathbf{m})^2(\alpha^2-1)\right)^2}, \text{kde } \mathrm{H}^+(\mathrm{x}) = \begin{cases} 1: \mathrm{x} > 0 \\ 0: \mathrm{x} \leq 0 \end{cases} \qquad \textbf{(5)}$$

Pro GGX existuje také anizotropní varianta, jíž tvar je v rovnici 6. Parametry α_t a α_b značí, že drsnost povrchu je vzhledem k tangentě a bitangentě různá. To platí například pro broušené kovy. Pokud se α_t a α_b rovnají, pak je funkce totožná s izotropní verzí.

$$\mathrm{D}(\mathbf{m}) = \frac{\mathrm{H}^+(\mathbf{n}\cdot\mathbf{m})}{\pi\alpha_\mathrm{t}\alpha_\mathrm{b}\left(\frac{(\mathbf{t}\cdot\mathbf{m})^2}{\alpha_\mathrm{t}^2}+\frac{(\mathbf{b}\cdot\mathbf{m})^2}{\alpha_\mathrm{b}^2}+(\mathbf{n}\cdot\mathbf{m})^2\right)^2} \qquad \textbf{(6)}$$

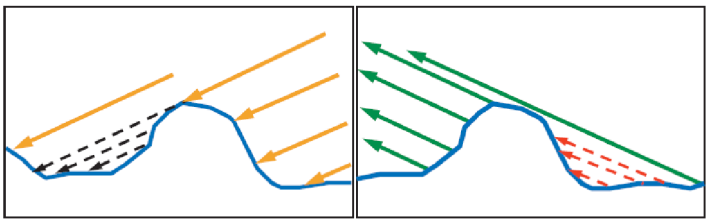


Obr. 7 Srovnání GGX DFN (dole) a Beckmann DFN (nahoře). Převzato z Real-Time Rendering [4].

4.1.2 Funkce zastínění a maskování

DFN, intuitivně řečeno, popisuje hustotu pravděpodobnosti mikroplošek, jichž normály se rovnají vektoru **m**. Je ovšem nutné počítat s tím, že ne všechny takové mikroplošky se budou ve skutečnosti podílet na odrazu světla ze směru **l** do směru **v**, protože paprsek bude buďto zastíněn nebo maskován. Oba termíny

označují situaci, kdy existují mikroplošky s danou normálou **h**, ale nejsou viditelné ze směru **l** (zastínění) a směru **v** (maskování), viz Obr. 8.



Obr. 8 Vizualice zastínění (nalevo) a maskování (napravo). Převzato z Real-Time Rendering [4].

Funkce zastínění a maskování $G(\mathbf{l}, \mathbf{v}, \mathbf{m})$ (zkráceně funkce G) tedy udává podíl mikroplošek, které jsou viditelné jak ze směru **l**, tak ze směru **v** [20]. Existuje více různých G funkcí, jako například Cook-Torrance [13], ale podle Erica Heitze [21] je nejlepší variantou Smithova G funkce [22].

Smithova G funkce má tvar $G(\mathbf{l}, \mathbf{v}, \mathbf{m}) = G_1(\mathbf{v}, \mathbf{m})G_1(\mathbf{l}, \mathbf{m})$, kde funkce $G_1(\mathbf{v}, \mathbf{m})$ řeší maskování a $G_1(\mathbf{l}, \mathbf{m})$ zastínění. Obecný tvar funkce G_1 udává rovnice 7. Funkce Λ musí být odvozena zvlášť pro každou DFN. Její tvar funkce pro GGX je uveden v rovnici 8 [20].

$$G_1(\mathbf{x}, \mathbf{m}) = \frac{H^+(\mathbf{x} \cdot \mathbf{m})}{1 + \Lambda(\mathbf{x})}, \text{ kde } H^+(x) = \begin{cases} 1: x > 0 \\ 0: x \leq 0 \end{cases} \quad \textbf{(7)}$$

$$\Lambda(x) = \frac{-1 + \sqrt{1 + \alpha^2 (\tan \theta_x)^2}}{2} \quad \textbf{(8)}$$

Je zde také uvedena verze podle Briana Karise [23], která místo funkce tangens využívá trigonometricky identických výrazů.

$$G_1(\mathbf{x}, \mathbf{m}) = \frac{2(\mathbf{n} \cdot \mathbf{x}) H^+(\mathbf{x} \cdot \mathbf{m})}{(\mathbf{n} \cdot \mathbf{x}) + \sqrt{\alpha^2 + (1 - \alpha^2)(\mathbf{n} \cdot \mathbf{x})^2}} \quad \textbf{(9)}$$

Eric Heitz doporučuje v praxi využívat přesnější tzv. výškově-korelovanou funkci zastínění a maskování. Ta vychází z toho, že čím výše se bod vyskytuje na mikropovrchu, tím nižší je pravděpodobnost, že bude zastíněn nebo maskován. Její tvar je následující:

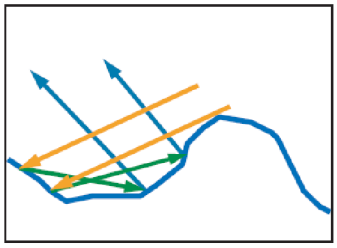
$$G(\mathbf{l},\mathbf{v},\mathbf{m}) = \frac{H^+(\mathbf{v}\cdot\mathbf{m})H^+(\mathbf{l}\cdot\mathbf{m})}{1+\Lambda(\mathbf{v})+\Lambda(\mathbf{l})} \quad \textbf{(10)}$$

Výškově-korelovanou G funkci je vhodné vyjádřit ve tvaru „visibility“ funkce (viz 4.1). Po dosazení a aplikování trigonometrických identit se vzorec zjednoduší, viz rovnice 11.

$$V(\mathbf{l},\mathbf{v},\mathbf{h}) = \frac{0.5}{(\mathbf{n}\cdot\mathbf{l})\sqrt{\alpha^2+(\mathbf{n}\cdot\mathbf{v})^2(1-\alpha^2)}+(\mathbf{n}\cdot\mathbf{v})\sqrt{\alpha^2+(\mathbf{n}\cdot\mathbf{l})^2(1-\alpha^2)}} \quad \textbf{(11)}$$

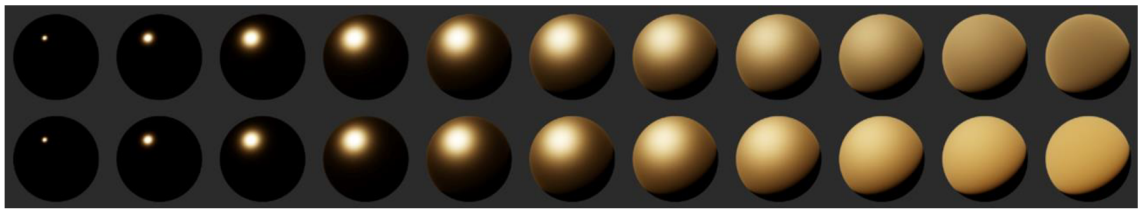
4.1.3 Vícenásobný odraz

Popisovaný spekulární model je založen na předpokladu, že se paprsek světla odrazí od mikroplošky a je buďto maskován nebo nikoliv. V realitě ale může také nastat situace, kdy se paprsek odrazí od několika mikroplošek, a nakonec se odrazí do směru **v**, viz Obr. 9. Taková situace je známá jako vícenásobný odraz (multiscattering).



Obr. 9 Vícenásobný odraz od mikroplošek.
Převzato z Real-Time Rendering [4].

Vícenásobný odraz je tudíž v rozporu s popisovaným modelem. Důsledkem je nedostatečné množství odrazu světla u drsných materiálů, zejména kovů. Řešením vícenásobného odrazu se zabývali např. Kulla a Conty [19]. Porovnání základního modelu a jejich modelu s vícenásobným odrazem je vidět na Obr. 10. Novější řešení nabízí Fdez-Agüera [24].



Obr. 10 Srovnání modelu bez vícenásobného odrazu (nahoře) a modelu s vícenásobným odrazem (dole).
Převzato z Physically Based Rendering in Filament [14].

4.1.4 Fresnelova funkce

Fresnelova funkce (dále funkce F) slouží k výpočtu množství odraženého světla tak, jak bylo popsáno v kapitole 2.1. Jak bylo řečeno, použití plných Fresnelových rovnic by bylo pro potřeby real-time aplikací zbytečně výpočetně náročné. Pro aproximaci těchto rovnic se nejčastěji používá Schlickova aproximace [25]:

$$\mathrm{F}(\mathbf{v},\mathbf{h}) = F_0 + (1 - F_0)\big(1 - (\mathbf{v}\cdot\mathbf{h})\big)^5 \qquad \textbf{(12)}$$

F_0 je vlastnost materiálu – udává odrazivost v případě, že je směr osvětlení **l** kolmý na povrch (θ = 0). V realitě se F_0 mění s vlnovou délkou světla, což opět znamená, že je v real-time grafice reprezentována jako RGB hodnota. F_0 je možné vypočítat pomocí indexu lomu daného materiálu pomocí této rovnice:

$$F_0 = \left(\frac{n_1 - n_2}{n_2 + n_2}\right)^2 \qquad \textbf{(13)}$$

kde n_1 je index lomu materiálu a n_2 typicky index lomu vzduchu. Podle rozsahu F_0 se materiály dělí na 3 skupiny: nevodiče (dielektrika), vodiče (kovy) a polovodiče. Nevodiče mají typicky velmi nízké F_0 – přibližně 0.04, viz Tabulka 1. Mezi výjimky patří například diamant, jehož F_0 je cca 0.13 až 0.2. Hodnoty F_0 dielektrik nejsou uvedeny v RGB, protože hodnoty jsou pro jednotlivé barevné kanály většinou totožné. Naopak kovy se vyznačují vysokými F_0 hodnotami v rozsahu od 0.5 do 1.0, viz Tabulka 2. F_0 hodnoty polovodičů se pohybují mezi dielektriky a kovy. V praxi je ale zobrazování takových materiálů poměrně vzácné.

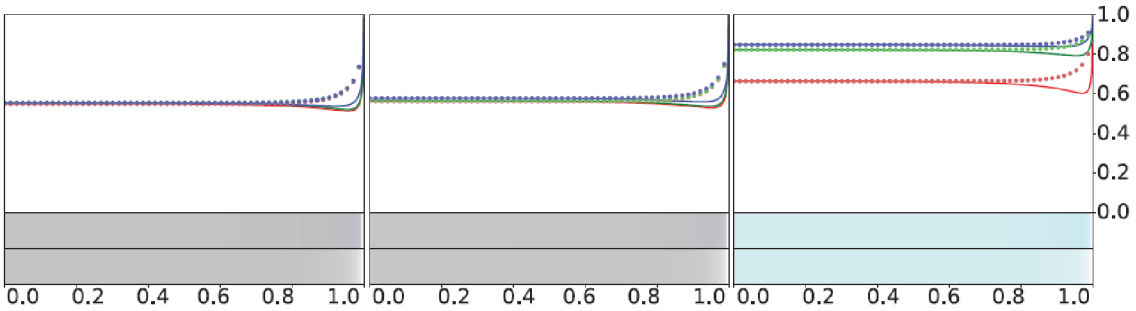
Tabulka 1 F_0 hodnoty pro vybraná dielektrika. Převzato z Real-Time Rendering [4]

Materiál	F_0
Voda	0.02
Kůže	0.028
Tkaniny	0.04 – 0.056
Plasty	0.04 – 0.05

Tabulka 2 F_0 hodnoty pro vybrané kovy. Převzato z Real-Time Rendering [4]

Materiál	F_0
Měď	(0.955, 0.638, 0.538)
Zinek	(0.664, 0.824, 0.850)
Zlato	(1.000, 0.782, 0.344)

Přesnost Schlickovy aproximace je pro většinu materiálů dostačující, avšak pro některé kovy se může od Fresnelových rovnic mírně lišit. Jak lze vidět na Obr. 11, pro hodnoty okolo 55° úhlu θ_i se množství odraženého světla pro kovy mírně snižuje. Na první pohled jde o značnou nepřesnost. Naty Hofman nicméně podotýká, že tato nepřesnost většinou postihne poměrně malé množství pixelů – ty, které se vyskytují na okrajích zobrazovaných objektů [26]. Podle Natyho Hofmana je také vhodnější v real-time aplikacích používat Schlickovu aproximaci a přesné Fresnelovy rovnice využívat pouze ve spojení se spektrálním renderováním.



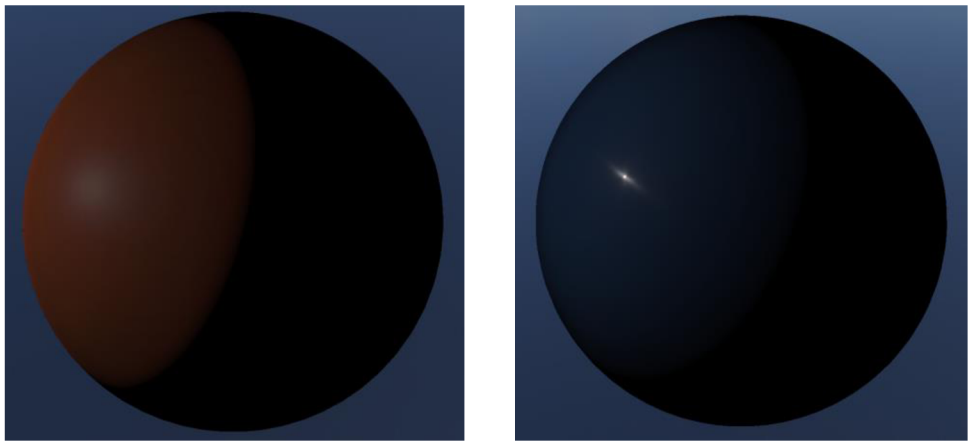
Obr. 11 Porovnání Schlickovy aproximace (tečkované křivky) s Fresnelovými rovnicemi (plné křivky) pro tři různé kovy (zleva): chrom, železo, zinek. Osa x je sin θ_i.
Převzato z Real-Time Rendering [4].

Pro aplikace, kde je důležité kovy přesně zobrazit existuje řešení od Laurenta Belcoura [27], které je vhodné pro real-time aplikace. Autoři navrhují Fresnelovy rovnice aproximovat pomocí lineární kombinace 4 křivek a zároveň navrhují vhodnou parametrizaci.

4.2 Difusní BRDF modely

Jak bylo řečeno v kapitole 2.2, difusní složka BRDF aproximuje lokální podpovrchový rozptyl. Nejjednodušším difusním modelem je tzv. Lambertovská BRDF, která určuje velikost difusní složky jako konstantní: $f_{diffuse} = \frac{albedo}{\pi}$. Lambertovská BRDF vyplývá z představy, že se světlo uvnitř materiálu rozptýlí náhodně, a tudíž je vyzářeno rovnoměrně do všech směrů. Ačkoliv tato představa není zcela korektní, Lambertovský model je pro mnohé aplikace dostatečně přesný. Brian Karis z Epic Games také podotýká, že jednoduchost Lambertovského modelu je ve skutečnosti výhodou, protože zjednodušuje implementaci osvětlovacích technik, jako například osvětlení na základě obrazu [28].

Lambertovský odraz není ve skutečnosti zcela přesný. Stejně jako spekulární složka je difusní složka ovlivněna drsností povrchu. Některé drsné materiály jsou typické svými retroreflexními vlastnostmi [29], což se může projevovat vizuálním „zploštěním“ zobrazovaného objektu. Tyto retroreflexní charakteristiky zachycuje např. Oren-Nayar model [15], který je založen na mikroploškovém modelu. Naopak některé hladké difusní materiály způsobují vizuální „zaoblení“, viz následující obrázek:



a) red-plastic (MERL) b) acrylic-blue (MERL)

Obr. 12 Dva různé difusní materiály. Vlevo drsný, vpravo hladký. Vlastní tvorba pomocí dat z MERL [8].

Podle Brenta Burleyho z Disney, který porovnal některé difusní modely [29] je však Oren-Nayar model nepřesný v porovnání s MERL materiály. Burley navrhuje vlastní difusní model, tzv. Disney Diffuse, který je spíše empirický, ale dobře napodobuje některé vlastnosti MERL materiálů. Tvar Disney difusního modelu:

$$f_{diffuse} = \frac{albedo}{\pi} F(\mathbf{n}, \mathbf{l}) F(\mathbf{n}, \mathbf{v}) \quad \textbf{(14)}$$

kde $F(\mathbf{x}, \mathbf{y})$ je upravená Schlickova funkce:

$$F(\mathbf{x}, \mathbf{y}) = 1 + (F_{D90} - 1)(1 - (\mathbf{x} \cdot \mathbf{y}))^5 \quad \textbf{(15)}$$

kde

$$F_{D90} = 0{,}5 + 2\alpha\,(\mathbf{l} \cdot \mathbf{h})^2 \quad \textbf{(16)}$$

Sébastien Lagarde z Electronic Arts se na rozdíl od Epic Games rozhodl v herním enginu Frostbite nepoužít Lambertovskou BRDF [18]. Místo toho používá verzi Disney difusního modelu modifikovanou tak, aby neporušovala zákon zachování energie. Zajímavé řešení difusní složky pochází ze hry Call of Duty: WWII [30]. Autoři nejprve simulovali mikroploškový difusní model a poté ho aproximovali analytickými rovnicemi. Výsledný model je v podstatě retroreflexní BRDF sečtená s interpolací mezi drsnou a hladkou difusní BRDF.

4.3 Modely pro vícevrstvé materiály

Prozatím byla věnována pozornost jednovrstvým materiálům. V reálném světě se ale vyskytuje poměrně mnoho objektů, které mají na svém povrchu více vrstev, které mohou významně ovlivňovat jejich vzhled. Příkladem mohou být lakované karosérie aut, plechovky od nápojů, hudební nástroje apod.

Renderování vícevrstvých materiálů je mnohem složitější než jednovrstvých, protože je potřeba uvažovat šíření světla napříč vrstvami. Pokaždé, když pomyslný paprsek světla prochází rozhraním mezi vrstvami, rozdělí se na odraženou a lomenou část. Při každém kontaktu světla s jednotlivými rozhraními mezi vrstvami by tudíž bylo potřeba vyhodnotit BSDF (bidirectional scattering distribution function). Z těchto důvodů je renderování vícevrstvých materiálů v reálném čase problematické. S poměrně novým řešením přišel opět Laurent Belcour, který navrhuje výsledný odraz světla vypočítat pomocí několika mikroploškových BRDF s GGX distribucí [31].

V dnešní době ovšem ještě není renderování mnohovrstvých materiálů v real-time aplikacích tolik rozšířené. Typicky je umožněno renderovat jednu přidanou vrstvu, která představuje vrstvu laku. Této vrstvě se říká „clearcoat“ [14, 29]. Renderování clearcoat vrstvy je většinou založeno spíše na ad-hoc řešeních. Odraz od vrstvy laku je modelován pomocí podobné mikroploškové BRDF jako od spodní vrstvy. Je vhodné snížit intenzitu světla odraženého od spodní vrstvy úměrně k množství světla odraženého od clearcoat vrstvy.

5 Osvětlovací model

Doposud bylo popisováno pouze modelování chování světla při dopadu na povrch či rozhraní, tj. matematický postup popisující, jakým způsobem objekty složené z určitých materiálů odráží světlo. Pro aplikování tohoto modelu je ale nezbytné definovat také model osvětlení. Osvětlovací model určuje pozici zdrojů světla, jejich vlastnosti a popisuje způsob, jakým jsou jednotlivé objekty osvětleny. V následujících podkapitolách budou popsány různé typy zdrojů světla.

5.1 Bodové zdroje světla

Bodové zdroje jsou nejjednodušší a dalo by se říci nejrozšířenější typy světelných zdrojů v real-time aplikacích. Bodový zdroj světla je popsán intenzitou světla a bodem v prostoru, ze kterého je světlo vyzařováno. Bodové zdroje mají nekonečně malý objem a tudíž nejsou realistické. Přesto jsou používány, a to zejména kvůli nízkým požadavkům na výkon.

Výpočet rovnice odrazu pro bodové zdroje se zjednoduší na diskrétní tvar:

$$L_{out}(\mathbf{p}, \mathbf{v}) = \sum f_{brdf}(\mathbf{p}, \mathbf{v}, \mathbf{l_{in}}) L_{in}(\mathbf{p}, \mathbf{l_{in}}) |\cos \theta_i| \qquad \textbf{(17)}$$

kde $L_{in}(\mathbf{p}, \mathbf{l_{in}})$ popisuje intenzitu světla pocházející od určitého bodového zdroje. Ta se řídí zákonem převrácených čtverců, tj. intenzita $\propto \frac{1}{\text{vzdálenost}^2}$. Častým problémem se zákonem převrácených čtverců je, že pro nulovou vzdálenost se intenzita rovná nekonečnu, což při renderování způsobuje značné potíže. To se dá vyřešit například přidáním konstanty do jmenovatele [28]: intenzita $\propto \frac{1}{\text{vzdálenost}^2+\varepsilon}$.

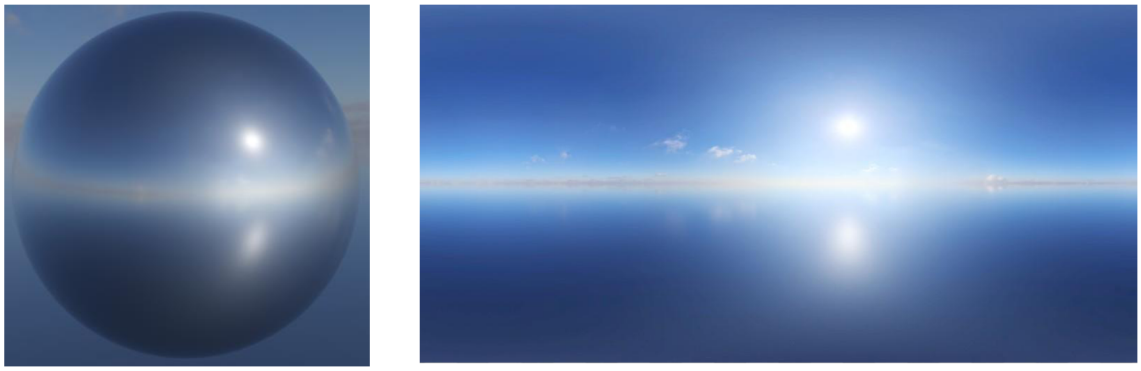
5.2 Plošné zdroje světla

Plošně zdroje světla na rozdíl od bodových reprezentují realistické světelné zdroje. Výpočet osvětlení z obecných plošných zdrojů je však pro real-time aplikace příliš výpočetně náročný, protože je potřeba vyřešit rovnici odrazu, která se pro obecné plošné zdroje světla řeší například metodami Monte-Carlo. Typy plošných zdrojů jsou proto často limitovány na útvary, pro které je možné výpočet aproximovat - například koule nebo válce [28]. Existuje také poměrně nová technika s názvem „Linearly Transformed Cosines“ (LTC) [32], která umožňuje osvětlení polygonálními zdroji světla v reálném čase. Původní verze LTC podporovala pouze BRDF založenou na izotropní GGX distribuci. Novější verze umožňuje tuto techniku používat také s anizotropní GGX distribucí [33].

5.3 Osvětlení na základě obrazu (Image-Based Lighting)

Osvětlení na základě obrazu, anglicky „Image Based Lighting“, zkráceně IBL je technika, která k osvětlení objektů využívá tzv. mapu prostředí. Mapa prostředí je funkce, která pro každý směrový vektor z jednotkové sféry určuje intenzitu

osvětlení z daného směru. Tato funkce je často reprezentovaná pomocí cubemapy, ve které každý texel určuje intenzitu osvětlení. IBL je poměrně jednoduchá technika, pomocí které lze značně obohatit vizuální kvalitu zobrazovaných objektů, viz Obr. 13. Typicky se výpočet IBL liší pro difusní a spekulární složku.



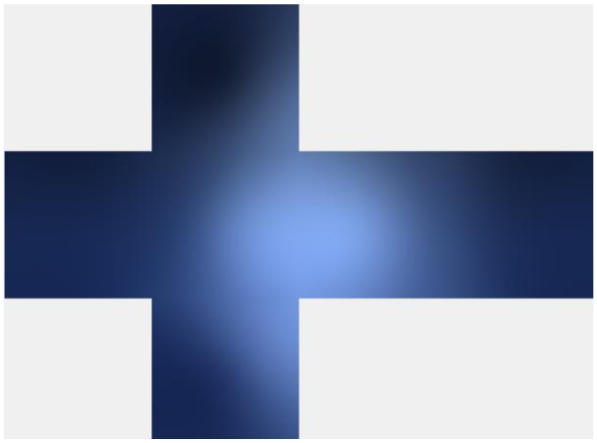
Obr. 13 Ukázka IBL. Zobrazený objekt (nalevo), mapa prostředí (napravo). Vlastní tvorba, autor mapy prostředí je Jarod Guest z Polyhaven.

5.3.1 Difusní složka – Irradiance Map

Výpočet difusní složky se řídí rovnicí odrazu. Pokud se jako difusní BRDF použije Lambertovská BRDF, pak se integrál z rovnice odrazu výrazně zjednoduší:

$$L_{out}(\mathbf{v}) = \frac{\text{albedo}}{\pi} \int_{\Omega} L_{in}(\mathbf{l_{in}})(\mathbf{n} \cdot \mathbf{l}) d\mathbf{l_{in}} \qquad \textbf{(18)}$$

Tento integrál je možné jednoduše vyřešit pomocí metody Monte-Carlo. Bylo by však neefektivní integraci provádět při běhu pro každý pixel zvlášť. Proto se integrál typicky přepočítá a uloží do tzv. Irradiance Mapy (mapa ozáření), z které se při renderování načte příslušný element. Irradiance Mapu je možné uložit jako texturu, viz Obr. 14. Efektivněji lze Irradiance Mapu uložit jako koeficienty sférických harmonických funkcí [34].



Obr. 14 Irradiance Mapa (mapa ozáření).
Vlastní tvorba.

5.3.2 Spekulární složka

Výpočet spekulární složky je podobně jako výpočet difusní složky řešením rovnice odrazu. V té už ale na rozdíl od difusní složky hraje roli spekulární BRDF. Pro spekulární BRDF není schůdné předem exaktně vypočítat všechny možnosti, protože je to funkce mnoha parametrů. Proto se pro předvýpočet spekulární složky často používá tzv. „Split-Sum" aproximace od Briana Karise [28], která díky několika zjednodušením umožňuje spekulární BRDF předvypočítat a uložit do textur. Tato technika je založena na vzorkování podle důležitosti:

$$L_{out}(\mathbf{v}) = \int_{\Omega} f_{brdf}(\mathbf{v}, \mathbf{l_{in}}) L(\mathbf{l_{in}})(\mathbf{n} \cdot \mathbf{l}) d\mathbf{l_{in}} \approx \frac{1}{N} \sum_{k=1}^{N} \frac{f_{brdf}(\mathbf{v}, \mathbf{l_k}) L(\mathbf{l_k})(\mathbf{n} \cdot \mathbf{l_k})}{p(\mathbf{l_k}, \mathbf{v})} \qquad \textbf{(19)}$$

Karis navrhuje tuto sumu aproximovat rozdělením na 2 části (viz detailní odvození podle Sébastiena Lagarda[18]):

$$L_{out}(\mathbf{v}) \approx \left(\frac{1}{N} \sum_{k=1}^{N} \frac{F(\mathbf{v}, \mathbf{h}) G(\mathbf{l_k}, \mathbf{v}, \mathbf{h})(\mathbf{v} \cdot \mathbf{h})}{(\mathbf{n} \cdot \mathbf{v})(\mathbf{n} \cdot \mathbf{h})} \right) \left(\frac{1}{\sum_{k}^{N} (\mathbf{n} \cdot \mathbf{l_k})} \sum_{k=1}^{N} L(\mathbf{l_k}) (\mathbf{n} \cdot \mathbf{l_k}) \right) \qquad \textbf{(20)}$$

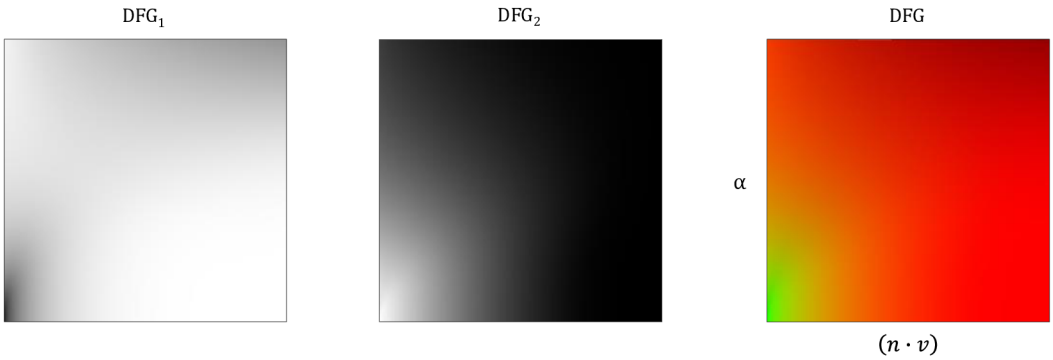
kde první suma je tzv. „DFG" část a druhá suma tzv. „LD" část. Intuitivně by se dalo říci, že DFG reprezentuje BRDF a LD reprezentuje osvětlení z mapy prostředí. Za předpokladu, že je použita Schlickova aproximace Fresnelovy funkce se DFG dále rozdělí na 2 části:

$$DFG = F_0 DFG_1 + F_{90} DFG_2 = F_0 \frac{1}{N} \sum_{k}^{N} (1 - F_c) G_{vis} + \frac{1}{N} \sum_{k}^{N} F_c G_{vis} \qquad \textbf{(21)}$$

kde

$$F_c = \left(1 - (\mathbf{v} \cdot \mathbf{h})\right)^5, G_{vis} = \frac{G(\mathbf{l_k}, \mathbf{v}, \mathbf{h})}{(\mathbf{n} \cdot \mathbf{v})(\mathbf{n} \cdot \mathbf{h})} \text{ a } F_{90} = 1 \qquad \textbf{(22)}$$

Jednotlivé části DFG_1 a DFG_2 se spočítají a uloží do textury, která slouží jako vyhledávací tabulka indexovaná $(\mathbf{n} \cdot \mathbf{v})$ a α, viz Obr. 15. Výpočet funkcí $F(\mathbf{v}, \mathbf{h})$ a $G(\mathbf{l_k}, \mathbf{v}, \mathbf{h})$ v DFG složce vyžaduje pohledový vektor $\mathbf{v}$, který při předvýpočtu samozřejmě není znám. Možným řešením je zvolit libovolnou normálu a vektor $\mathbf{v}$, spočítat tak, aby vyhovoval hodnotě $(\mathbf{n} \cdot \mathbf{v})$ [14].



Obr. 15 DFG textura.
Vlastní tvorba.

LD složka reprezentuje osvětlení z mapy prostředí. Suma uvnitř LD složky se spočítá pro více úrovní drsnosti, což je podle Karise známý postup používaný v herním průmyslu. Důvodem je to, že na materiály s vyšší drsností má vliv osvětlení z mnohem většího prostorového úhlu, kdežto na materiály s nízkou drsností má vliv osvětlení z velice úzkého prostorového úhlu. Prakticky to znamená, že je použito vzorkování podle důležitosti, kde se vzorky vytvářejí podle distribuční funkce normál dané BRDF. Takto se vytvoří několik textur, každá pro různou úroveň drsnosti. Textury pro vyšší drsnosti většinou neobsahují vysokofrekvenční data, a proto je možné je bez větší ztráty kvality uložit do jednotlivých MIP úrovní jedné textury, viz Obr. 16.

Pro výpočet vzorkování podle důležitosti pro GGX distribuci je potřeba znát pohledový vektor, který při předvýpočtu samozřejmě určit nelze. Zároveň kvůli paměťové náročností není schůdné provést předvýpočet pro různé hodnoty pohledového vektoru. Z těchto důvodu navrhuje Karis předvýpočet zjednodušit tím, že se pohledový vektor vždy bude rovnat normále **n**, což bohužel v některých situacích vede k nerealistickým výsledkům.

Obr. 16 MIP úrovně LD textury.
Vlastní tvorba.

6 Formát GLTF

Existuje mnoho souborových formátů pro ukládání 3D modelů, jako například Wavefront OBJ, FBX, Collada, STL atd… Žádný z nich však nebyl vytvořen s podporou PBR materiálů. Proto je v této práci použit docela nový formát GLTF 2.0 (dále GLTF) [35], který naopak PBR materiály standardně podporuje. GLTF je velice flexibilní formát, který by měl podle tvůrců fungovat jako „JPEG 3D formátů“ [36].

GLTF není pouze formátem pro ukládání jednotlivých modelů, ale umožňuje ukládat celé scény, včetně pozice kamer, světel, animací atd… Samotný formát GLTF je rozdělen na 2 části: binární data (textury, geometrie) a popis souboru, který je uložen ve formátu JSON. Existuje několik způsobů, kterým lze tyto data uložit – buďto do JSON souboru s příponou „.gltf“ nebo do binárního souboru s příponou „.glb“.

Důležitou vlastností GLTF je, že struktura GLTF scény se skládá z uzlů, které jsou uspořádány do stromové hierarchie. Každý uzel může reprezentovat jiný objekt, například 3D geometrii, kameru, „klouby“ pro animaci, atd… Vyčerpávající popis formátu GLTF však není možný, protože se jedná o velice flexibilní formát, který podporuje mnoho různých možností. Proto bude nadále věnována pozornost specifikaci materiálů, což je část relevantní pro tuto práci.

6.1 PBR materiály v GLTF

Materiálový model v GLTF je od základů navržen tak, aby byl dobře použitelný s mikroploškovým BRDF modelem. Ten má ve své nejjednodušší formě 3 materiálové parametry: c_{diff} (difusní albedo), α (drsnost povrchu) a F_0 (hodnota Fresnelovy funkce pro úhel 0). Rozšiřujícím parametrům budou věnovány následující podkapitoly. Základním PBR formátem v GLTF je tzv. „Metallic-Roughness“ model. Také existuje rozšíření pro podporu podobného formátu „Specular-Glossiness“.

Metallic-Roughness definuje celkem 3 parametry: *základní barva*, *kovovost* (Metallic) a *drsnost* (Roughness). Mapování těchto parametrů na BRDF parametry je následující. *Drsnost* přesně odpovídá α. Určení c_{diff} a F_0 závisí na hodnotě *kovovosti* a *základní barvy*, viz Tabulka 3. Pro hodnoty *kovovosti* mezi 0 a 1 jsou c_{diff} a F_0 interpolovány.

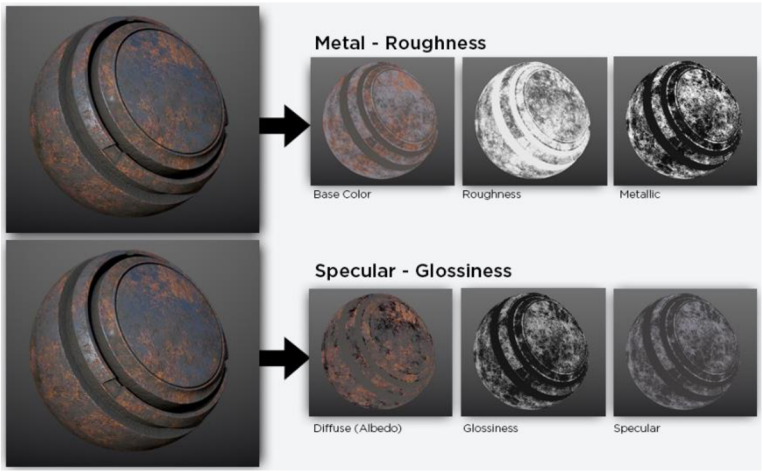
Tabulka 3 Určení c_{diff} a F_0 podle kovovosti.

Kovovost	1	0
F_0	*základní barva*	0.04
c_{diff}	0	*základní barva*

Specular-Glossiness model je expresivnější než Metallic-Roughness, používá také 3 parametry: *lesk* (Glossiness), *specular* a *diffuse* (albedo). Mapování parametrů je jednodušší než u Metallic-Roughness. Lesk je inverzní hodnota k α, tudíž $\alpha = 1 -$ lesk. *Specular* přímo udává F_0. Výpočet c_{diff} je podle následující rovnice:

$$c_{diff} = diffuse\big(1 - \max(specular.r, specular.g, specular.b)\big) \qquad \textbf{(23)}$$

Metallic-Roughness a Specular-Glossines sice určují stejné BRDF parametry, ale jsou mezi nimi mírné rozdíly. První je ten, že Specular-Glossiness umožňuje určit F_0 i pro dielektrika, kdežto Metallic-Roughness určuje F_0 pouze pro kovy. To může být užitečné při renderování dielektrik s neobvyklým indexem lomu jako jsou například drahokamy. Nevýhodou Specular-Glossiness je, že je nutné uložit *specular* jako RGB hodnotu, což vede k vyšší paměťové náročnosti než Metallic-Roughness. Metallic-Roughness model navíc umožňuje sloučení kovovosti a drsnosti do jedné textury. Z těchto důvodu doporučují autoři formátu GLTF pro nové soubory používat výhradně Metallic-Roughness. Pokud je potřeba specifikovat F_0, lze využít rozšíření pro index lomu. Srovnání Metallic-Roughness a Specular-Glossiness lze vidět na následujícím obrázku:



Obr. 17 Srovnání Metallic-Roughness a Specular-Glossiness. Převzato z PBR Guide by Allegorithmic [1].

6.1.1 Clearcoat

Clearcoat je rozšířením standardního materiálového modelu, viz kapitola 4.3. Pro formát GTLF je definována jako rozšíření [37] základní specifikace. Pro definici clearcoat vrstvy lze použít několik parametrů: clearcoatFactor, clearcoatRoughness a clearcoatNormalTexture. ClearcoatFactor definuje intenzitu odrazu světla od vrstvy laku. ClearcoatRoughness je drsnost, stejně jako u základní vrstvy materiálu. Oba tyto parametry mohou být také definovány texturou. ClearcoatNormalTexture definuje normálovou mapu nezávisle na normálové mapě základní vrstvy.

6.1.2 Anizotropie

Anizotropní materiály zatím nejsou v GLTF podporovány, ale jsou plánovány jako rozšíření[38]. Proto je v této práci použit anizotropní parametr od Imageworks [19]. Anizotropní materiál je v podstatě stejný jako základní materiál, ale vyžaduje 2 parametry drsnosti: α_t (drsnost podél tangenty) a α_b (drsnost podél bitangenty). Podle Imageworks je vhodné použít jeden parametr *anizotropie* v rozsahu od -1 do 1, podle kterého jsou α_t a α_b vypočítány následujícím způsobem:

$$\begin{aligned} \alpha_{\mathrm{t}} &= \alpha(1 + \text{anizotropie}) \\ \alpha_{\mathrm{b}} &= \alpha(1 - \text{anizotropie}) \end{aligned} \tag{24}$$

Dále je potřeba vyřešit, jak renderovat anizotropní materiály pomocí techniky IBL. Podobný předvýpočet jako u izotropních materiálů není možný kvůli vyššímu množství parametrů. Technika, která je v této práci použita je založena na vychýlení vektoru odrazu [17]. Nejedná se o korektní výpočet anizotropního odrazu, ale pouze o hrubou aproximaci, která vede k přijatelným výsledkům.

7 Návrh a implementace

Cílem práce je mimo jiné implementovat řešení demonstrující vykreslování 3D scény s využitím PBR metod pracující v reálném čase. Jako způsob řešení byla zvolena standardní desktopová aplikace fungující jak na OS Windows, tak na OS Linux. Aplikace bude umožňovat načítat a zobrazovat scény ve formátu GLTF, které budou osvětleny pomocí vybraných technik z kapitoly 5. Výpočet osvětlení bude využívat fyzikálně založené BRDF, pro jejichž výpočet by měly být využity materiálové parametry uložené v GLTF souboru. Pro implementaci zobrazovacích algoritmů bude využito vhodné API umožňující programování grafických karet. V následujících kapitolách budou popsány použité technologie, datové struktury pro načítání GLTF modelů, design rendereru a především implementace osvětlovacích metod.

7.1 Použité technologie

Pro implementaci byl zvolen programovací jazyk Rust. Jedná se o moderní, cca 10 let starý jazyk, jehož použití v oblasti grafických aplikací není obvyklé. Je

relativně nízko-úrovňový, podobně jako C++ nebo C, což je při využívání grafických API žádoucí. Na rozdíl od jazyků C/C++ ovšem slibuje bezpečnou práci s pamětí, tzn. při běžné práci s Rustem není možné způsobit známé chyby jako použití neplatného ukazatele, dvojité uvolnění alokované paměti apod. Rust zároveň nabízí zajímavé funkce jako například algebraické datové typy.

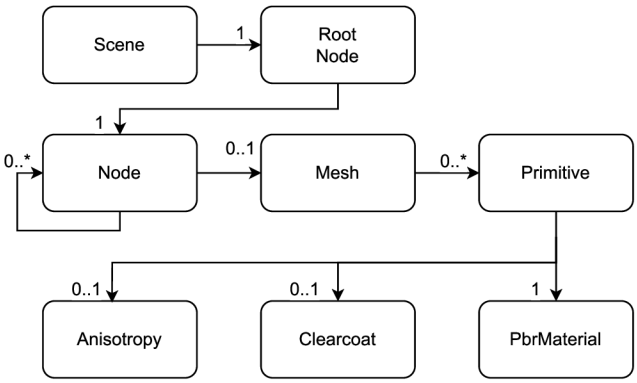
Pro samotné vykreslení 3D scény bylo použito API OpenGL v jeho nejnovější verzi 4.6. OpenGL dává programátorovi dostatečnou svobodu při implementaci grafických algoritmů, ale zároveň není tak složité jako Vulkan nebo Direct3D 12. Z novějších funkcí OpenGL byly použity například compute shadery.

Pro tvorbu GUI aplikace byla použita knihovna egui [39], která funguje podobně jako známá C++ knihovna ImGui [40]. O získávání vstupu z klávesnice a myši a tvorbu okna aplikace se stará známá knihovna SDL2 [41].

7.2 Načítání GLTF scén

Pro zjednodušení načítání GLTF souborů byla použita knihovna gltf-rs pro jazyk Rust. Každý GLTF soubor je v mé implementaci reprezentován strukturou scény – `Scene`[2], která obsahuje stromovou hierarchii uzlů (anglicky „`Node`“). Každý uzel může obsahovat 3D geometrii, uloženou ve struktuře `Mesh`. Formát GLTF dále dělí každý `Mesh` na několik struktur jménem `Primitive`, což jsou datové struktury, které obsahují 3D vrcholy s konkrétní topologií (trojúhelníky, úsečky apod.). Struktura `Primitive` je složena klasicky z vertex bufferu, index bufferu a ze struktur popisujících PBR materiály. Struktury `PbrMaterial`, `Clearcoat` a `Anisotropy` se odkazují na materiálové textury sdílené jedním GLTF souborem.

[2] Technicky vzato může každý GLTF soubor obsahovat více scén, ale v praxi jsou takové soubory vzácné.



Obr. 18 Diagram struktury GLTF scény. Vlastní tvorba.

GLTF je velmi flexibilní formát, který nabízí spoustu různých možností a parametrů. V této práci nebylo z časových důvodů možné implementovat všechny možnosti, které formát GLTF nabízí. Například jediná topologie, kterou implementovaný renderer podporuje je seznam trojúhelníků s index bufferem. Na první pohled by se mohlo zdát, že podpora jediné topologie bude silně limitující. Avšak při vývoji rendereru bylo v rámci testování použito mnoho modelů stažených z internetu a žádný z nich jinou topologii neobsahoval. Stejně tak je načítání limitováno na modely, které obsahují normály vertexů. Formát GLTF nevyžaduje, aby normály vertexů byly v souboru uloženy, ale modely bez předvypočítaných normál jsou opět vzácné.

Problém, který se při vývoji poměrně těžko řešil je kalkulace tangent definovaných ve vrcholech, které jsou potřeba například při mapování normál nebo při použití anizotropních BRDF. Formát GLTF nevyžaduje přítomnost tangent ve vrcholech. Na rozdíl od normál se tangenty v běžných GLTF souborech nevyskytují zdaleka tak často a je proto potřeba je při načítání modelu spočítat. Specifikace formátu GLTF vyžaduje, aby byly tangenty spočítány pomocí poměrně komplikované techniky MikkTSpace [42], pro kterou zatím neexistuje stabilní implementace v jazyce Rust. Proto byl výpočet tangent implementován pomocí jednoduchého algoritmu převzatého z [43]. Tento algoritmus bohužel není zdaleka tak robustní jako MikkTSpace, což může vést k chybám při vykreslování.

7.3 Renderer

Vytvořený renderer využívá standardní techniky programovatelné OpenGL pipeline, tedy vertex shadery, fragment shadery a několik compute shaderů použitých při předzpracování.

Renderování jednoho snímku se řídí jednoduchým algoritmem. Nejprve se nastaví globální OpenGL stav, jako například zapnutí z-testu, zapnutí backface culling apod[3]. Dále se aktualizují uniformní data pro shadery. Pro tento účel byly zvoleny tzv. „uniform buffer objects“ (UBO). Na rozdíl od klasických `uniform` parametrů mají UBO tu výhodu, že je nutné je aktualizovat pouze jednou a změna se projeví ve všech shader programech, které daný UBO využívají. Renderer využívá 3 hlavní UBO: `Ligthing` buffer pro uložení dat o světlech, `PbrMaterial` buffer obsahující materiálové parametry, a nakonec `Transforms buffer`, který ukládá transformační matice MVP. Následně jsou vykreslena světla, poté samotná GLTF scéna, a nakonec je vykreslen skybox (cubemapa).

O vykreslení GLTF scény se stará funkce `render_gltf_node`, která rekurzivně prochází strukturu scény, viz. Obr. 18. V případě, že narazí na uzel obsahující `Mesh`, vykreslí ho pomocí funkce `render_mesh`. Postup vykreslení Mesh se skládá z několika kroků. Prvně je nastavena modelová transformační matice objektu. Každý uzel totiž může obsahovat vlastní transformační matici indikující transformaci, která se má aplikovat pro všechny potomky. Následně jsou nastaveny příslušné textury, které `Mesh` využívá, je aktualizován UBO s materiálovými vlastnostmi, a nakonec je zvolen správný shader, pomocí kterého je geometrie vykreslena. Práce se shadery se ukázala být složitou složkou implementace, proto jí bude věnována následující kapitola.

7.4 Práce se shadery v OpenGL

Shadery tvoří nezbytnou část programovatelné pipeline. Práce s GLSL shadery je ovšem z hlediska dobrého softwarového inženýrství velmi náročná. Jedním z důvodů je absence systému pro moduly či balíčky. V běžném

[3] Důvodem nastavení OpenGL stavu při každém snímku, a ne pouze při startu programu je použití knihovny egui, která pro vykreslení taktéž používá OpenGL, čímž může mít na globální OpenGL stav vliv.

programovacím jazyce je možné kód rozdělit do více souborů a kód z jednoho souboru využít ve druhém. To je nezbytné ve chvíli, kdy například několik algoritmů využívá stejnou datovou strukturu. OpenGL žádnou takovou funkcionalitu nepodporuje. V situaci, kdy několik shaderů využívá tutéž datovou strukturu, je nutné strukturu nakopírovat do každého shaderu zvlášť. V takovém případě je při každé změně struktury nutné strukturu upravit v každém souboru, ve kterém je použita, což při vývoji vedlo k těžko dohledatelným chybám.

7.4.1 Balíčkový systém

Z těchto důvodů byl implementován vlastní balíčkový systém inspirovaný řešením z [44]. Pokud je potřeba v shaderu importovat kód z jiného souboru, použije se následující syntaxe:

```
{% include "structs/lighting.glsl" %}
```

Balíčkový systém je naprogramován pomocí šablonovací knihovny Tera [45]. Tera není použita pouze pro balíčkový systém, ale také pro jednoduché nastavování konstant. Jedním z náročných problému je synchronizace konstant mezi Rust kódem a GLSL kódem. Pokud se například změní číslo portu textury, je nutné toto číslo změnit jak v GLSL, tak v Rust kódu. Z důvodu vyhnutí se nutnosti aktualizovat data na dvou různých místech jsou tyto konstanty uloženy v separátním Rustovém modulu `shader_constants`, pomocí kterého Tera nastaví konstanty v GLSL kódu. Například deklarace textury v shaderu má následující tvar:

```
layout(binding = {{texture_ports.albedo}}) uniform sampler2D abledoTex;
```

Spuštění knihovny Tera probíhá při sestavení programu. Tera projde všechny zdrojové kódy shaderů a nahradí výrazy ve dvojitých složených závorkách za jejich reálné hodnoty. V případě příkazu `include` provede import specifikovaného souboru. Vytvořené shadery uloží do složky `shaders_stitched`.

7.4.2 Specializace shaderů

Druhým náročným problémem, který byl při implementaci řešen je specializace shaderů. Fragment shader využívající techniku PBR musí podporovat široké spektrum kombinací. Příkladem může být podpora různého množství textur. Základní materiálový model GLTF jich podporuje několik: albedo textura, metallic-

roughness textura, normálová mapa, ambient occlusion textura a emisní textura. Žádná z těchto textur však není povinná. Celkově 5 textur znamená 32 různých variant, se kterými musí shader umět pracovat. Pro řešení tohoto problému byly použity preprocesorové direktivy GLSL. V případě, že model obsahuje například normálovou mapu se do zdrojového kódu shaderu přidá direktiva `#define NORMAL_MAP`. Výpočet normálového vektoru fragmentu je poté při kompilaci shaderu specializován podle toho, zda je daná direktiva přítomna:

```
#ifdef NORMAL_MAP
    normal = getNormalFromMap(normalTex, ...);
#else
    normal = normalize(vsOut.normal);
#endif
```

Výpis kódu 1: Příklad specializace v GLSL kódu.

Na straně Rustu je nutné použité varianty shaderů uložit. Pro tento účel byla zvolena datová struktura HashMap. Hodnoty HashMapy jsou zkompilované shaderové programy. Klíče jsou generické struktury podporující trait[4] `ShaderDefines`. Tyto struktury definují data potřebná pro specializaci shaderu, např. použité textury pro PBR shader. Trait `ShaderDefines` vyžaduje, aby struktura implementovala metodu `fn defines(&self) -> Vec<&str>`, která umožní pro konkrétní instanci získat direktivy, které se mají do zdrojového kódu shaderu přidat. Například, pokud bude konkrétní struktura `PbrDefines` obsahovat normálovou mapu a albedo texturu, bude funkce `defines` vracet seznam `[“NORMAL_MAP“, “ALBEDO_MAP“]` zapsaný do zdrojového kódu specializovaného shaderu.

[4] Trait je v Rustu konstrukt, který umožňuje polymorfismus. Je to obdoba interface z jiných jazyků.

7.5 Implementace osvětlovacího modelu

Pro implementaci osvětlovacího modelu bylo zvoleno několik BRDF, 2 osvětlovací techniky a byla implementována 2 materiálová rozšíření:

- BRDF
    - Spekulární
        - Mikroploškový model se Schlickovou aproximací Fresnelovy funkce, GGX distribuční funkcí (jak izotropní, tak anizotropní) a Smithovou výškově-korelovanou G funkcí.
    - Difusní
        - Lambert.
        - „Frostbite“ difusní model.
        - „Call of Duty: WWII“ difusní model.
- Osvětlovací modely
    - Osvětlení bodovými zdroji světla.
    - Osvětlení na základě obrazu (IBL).
- Materiály
    - Analytické
        - Metallic-Roughness.
        - Clearcoat rozšíření.
        - Jednoduchý parametr anizotropie.
    - Naměřené
        - MERL BRDF databáze.
        - UTIA BRDF databáze.

Je nutno podotknout, že ne všechny BRDF a materiály jsou kompatibilní s oběma osvětlovacími technikami. Všechny materiály a BRDF lze využít s bodovými zdroji světla. Naopak osvětlení na základě obrázků (IBL) je v této implementaci limitováno na zmíněnou spekulární BRDF a Lambertovskou difusní BRDF. IBL je dále limitováno na analytické materiály s tím, že anizotropie je pouze hrubě aproximována.

Implementace samotných BRDF pomocí programovatelné OpenGL pipeline je přímočará – jde pouze o přepsání matematických výrazů do GLSL kódu. Například implementace GGX distribuce z rovnice 5 vypadá následovně:

```
float distributionGgx(float NoH, float roughness)
{
    float asq = roughness * roughness;
    float denom = (NoH * NoH) * (asq - 1.) + 1.;
    return (asq) / (PI * denom * denom);
}
```

Výpis kódu 2: Implementace GGX distribuce v GLSL.

7.5.1 Osvětlení bodovými zdroji světla

Postup použití fyzikálně založených BRDF pro osvětlení bodovými zdroji světla se od empirických BRDF nijak zásadně neliší. Pro každý zdroj světla je na základě vektoru pohledu, vektoru osvětlení a materiálových vlastností spočítána jak spekulární, tak difusní složka BRDF. Síla difusní složky se musí škálovat inverzní hodnotou kovovosti, tzn. $difuse = diffuse(1 - kovovost)$, protože kovy difusní složku absorbují. Dále je nutné škálovat difusní složku úměrně k síle spekulární složky (odraženého světla). Nejjednodušší způsob, kterým toho docílit je $difuse = diffuse(1 - Fresnel)$. Zde je nutné podotknout, že škálování inverzní hodnotou Fresnelovy funkce není zdaleka korektní a jde pouze o aproximaci. Finální hodnotu BRDF tedy tvoří spekulární složka a dvakrát škálovaná difusní složka.

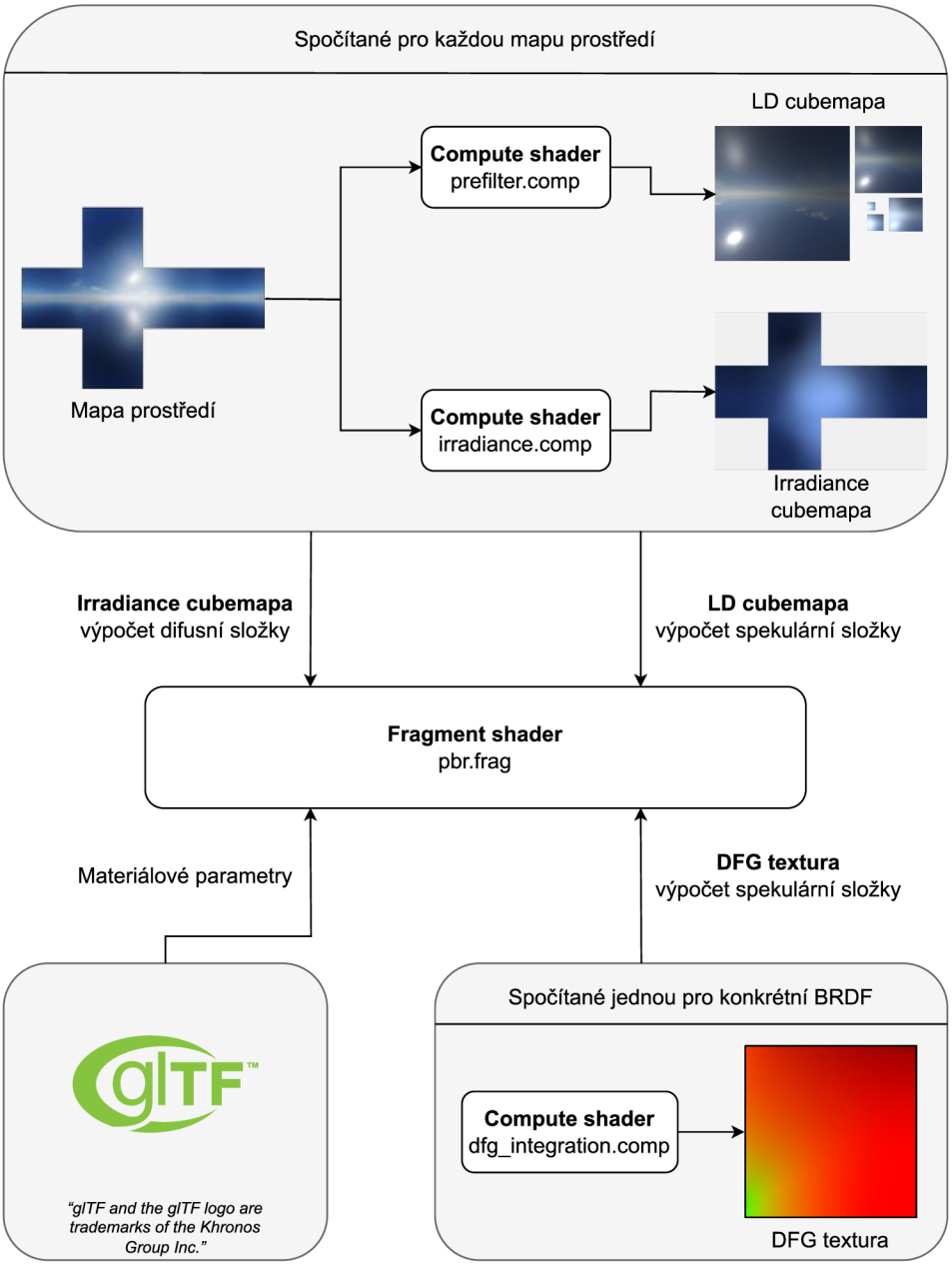
Pro rozšiřující materiálový model Clearcoat se musí postup výpočtu BRDF mírně upravit:

```
brdf = brdf * (1. - clearcoatFresnel) + clearcoatColor;
```

Proměnná `brdf` zde označuje hodnotu brdf základní materiálové vrstvy. Ta se opět škáluje inverzní hodnotu Fresnelovy funkce stejně jako tom bylo u výpočtu základní vrstvy. Poté je přidána BRDF vrstvy laku. Výpočet samotné Clearcoat BRDF závisí na poměrně vysokém množství parametrů a textur – GLTF je v tomto ohledu neobvykle flexibilní. Clearcoat vrstvu je možno ovlivnit několika parametry: drsnost, intenzita a normálová mapa. Drsnost a intenzitu lze navíc určit texturou. Detailní postup využití všech parametrů lze nalézt v GLTF dokumentaci [37].

7.5.2 Osvětlení na základě obrazu (IBL)

Mnohem komplexnější je implementace osvětlovacího modelu s použitím IBL. Schéma popisující použití IBL lze vidět na Obr. 19. Postup se dělí na 2 části: předzpracování a samotné vykreslení. Při předzpracování je nutné provést několik úkonů. Prvním je spočítání DFG textury. To je nutné provést pouze jednou při startu programu. Za druhé je potřeba vytvořit LD cubemapu a irradiance cubemapu. Při případné změně mapy prostředí se musí tento výpočet opakovat.



Obr. 19 Schéma zobrazovacího algoritmu využívající techniku IBL. Vlastní tvorba.

Při samotném vykreslování se na základě materiálových vlastností načtených z GLTF modelu a na základě zmíněných 3 textur spočte barva pixelu. Tuto úlohu má na starosti fragment shader. Výpočet spekulární složky vypadá následovně:

```
vec3 approximateSpecularIBL(...)
{
    // N - normála, V - vektor k pozorovateli.
    float NoV = clamp(dot(N, V), 0., 1.);
    vec3 reflectDir = 2 * NoV * N - V;

    const LD_ROUGHNESS_MIP_LEVELS = ... // Počet MIP úrovní LD cubemapy
    float mip = roughness * LD_ROUGHNESS_MIP_LEVELS;
    vec3 prefilteredColor = textureLod(ldTexture, reflectDir, mip).xyz;
    vec2 dfg = texture(brdfLut, vec2(sp.NoV, sp.roughness)).xy;

    vec3 specular = prefilteredColor * (f0 * EnvBRDF.x + EnvBRDF.y);
}
```

Výpis kódu 3: Výpočet spekulární složky IBL ve fragment shaderu, založeno na kódu z Real Shading in Unreal Engine 4 [28].

Pro načtení dat z LD cubemapy se používá explicitní MIP úroveň spočítaná podle drsnosti materiálu. Dále se načte element z DFG textury pomocí skalárního součinu $\mathbf{n} \cdot \mathbf{v}$ a drsnosti. Pomocí těchto složek je spočítána spekulární složka světla tak, jak popisují rovnice 20 a 21. Difusní složka je spočítána podobně jako v osvětlovacím modelu pro bodové zdroje světla s tím rozdílem, že osvětlení zde určuje Irradiance Mapa:

```
vec3 approximateDiffuseIBL(...)
{
    vec3 irradiance = texture(irradianceMap, N).xyz; // N - normála
    vec3 diffuse = irradiance * albedo * (1. - metalness);
}
```

Výpis kódu 4: Výpočet difusní složky IBL ve fragment shaderu, založeno na kódu z LearnOpenGL [46].

Problém nastává při snaze kombinovat difusní a spekulární složku IBL. Nabízí se jednoduché škálování difusní složky Fresnelovou funkcí, což nicméně může v některých případech vést k porušení zákona zachování energie. Tento starší způsob výpočtu spekulární složky navíc nebere v potaz vícenásobný odraz, viz kapitola 4.1.3. Ze zmíněných důvod byl v této práci použit novější model z článku „A Multiple-Scattering Microfacet Model for Real-Time Image-based Lighting“ [24], který využívá stejné textury, ale zároveň řeší problém vícenásobného odrazu a lépe balancuje množství energie mezí difusní a spekulární složkou. Kód pro tento model

je uveden ve Výpis kódu 5. Základní idea této techniky je, že vícenásobné odrazy budou spíše difusního charakteru, a proto je lze aproximovat pomocí Irradiance Mapy.

```
vec3 approximateIBLMultiscattering(...)
{
    // Roughness dependent fresnel
    vec3 Fr = max(vec3(1. - roughness), f0) - f0;
    vec3 kS = f0 + Fr * pow(1. - NoV, 5.);

    vec2 dfg = texture(brdfLut, vec2(NoV, roughness)).xy;
    vec3 FssEss = kS * dfg.x + dfg.y;

    const LD_ROUGHNESS_MIP_LEVELS = ... // Počet MIP úrovní LD cubemapy
    float mip = roughness * LD_ROUGHNESS_MIP_LEVELS;
    vec3 reflectDir = reflect(-V, N);
    vec3 radiance = textureLod(ldTexture, reflectDir, mip).xyz;
    vec3 irradiance = texture(irradianceMap, N).xyz;

    // Multiple scattering
    float Ess = dfg.x + dfg.y;
    float Ems = 1. - Ess;
    vec3 Favg = f0 + (1. - f0) / 21.;
    vec3 Fms = FssEss * Favg / (1. - (1. - Ess) * Favg);

    // Dielectrics
    vec3 Edss = 1 - (FssEss + Fms * Ems);
    vec3 kD = albedo * Edss;

    // Composition
    return FssEss * radiance + (Fms * Ems + kD) * irradiance;
}
```

Výpis kódu 5: Kompletní výpočet IBL s použitím modelu s vícenásobným odrazem, převzato z A Multiple-Scattering Microfacet Model for Real-Time Image Based Lighting [24].

7.5.2.1 Použití IBL s materiálovými rozšířeními

Renderování Clearcoat materiálů s použitím IBL je velice podobné jako jejich renderování s bodovými zdroji světla, tzn. přidá se jeden spekulární odraz, který je škálován inverzní hodnotou Fresnelovy funkce. Zajímavější je implementace anizotropních materiálů. V případě anizotropních materiálů není možné BRDF předzpracovat do podoby textury kvůli vyššímu množství parametrů. Proto byla využita již zmíněná aproximace založená na vychýlení vektoru odrazu [17]. Kód pro výpočet vychýlení vektoru byl převzat z dokumentace enginu Filament [14]. V případě renderování anizotropního materiálu je tedy pouze nutné vychýlit vektor `reflectDir` z Výpis kódu 5 nebo Výpis kódu 3.

7.5.2.2 Předzpracování IBL

Textury nutné pro použití IBL je možné vygenerovat pomocí nástrojů jako například IBLBaker [47] a načíst je při startu programu, nicméně pro herní enginy může být důležité tyto textury generovat za běhu – například při kontinuálních změnách mapy prostředí.

Všechny 3 textury je vhodné uložit ve formátu s plovoucí řádovou čárkou, zejména proto, že LD cubemapa a Irradiance cubemapa mohou obsahovat hodnoty větší než 1. U DFG textury je naopak důležitá přesnost. Podle Briana Karise je dostačující přesnost 16-bitových čísel. V implementovaném rendereru byla použita 32-bitová přesnost, což odpovídá OpenGL formátu `GL_RGBA32F`.

Jako první je potřeba načíst mapu prostředí a zkonvertovat ji do cubemap textury. Většina obrázků reprezentujících mapu prostředí je uložena pomocí ekvidistantní válcové projekce. Pro konverzi mezi ekvidistantní projekcí a cubemapou je použit compute shader, který je dle mých měření rychlejší než načítání jednotlivých stěn cubemapy z disku.

Dále je potřeba pro mapu prostředí spočítat Irradiance Mapu, viz rovnice 18. Integrál z rovnice 18 je vhodné zapsat pomocí sférických souřadnic. Po převedení do diskrétní formy vypadá rovnice následovně [46]:

$$L_{out}(\mathbf{v}) \approx \text{albedo}\,\frac{\pi}{n_1 n_2}\sum_{\phi_i=0}^{n_1}\sum_{\theta_i=0}^{n_2} L(\phi_i,\theta_i)(\mathbf{n}\cdot\mathbf{l})\sin\theta_i \qquad \textbf{(25)}$$

Pro výpočet této rovnice a uložení výsledku do Irradiance Mapy je opět použit compute shader. Každá invokace compute shaderu je identifikována proměnnou `gl_GlobalInvocationID`, jíž **xy** souřadnice určují souřadnice pixelu ve stěně cubemapy a souřadnice **z** určuje stěnu cubemapy. Algoritmus compute shaderu vypadá následovně:

```
load xyz coordinates from gl_GlobalInvocationID
calculate normal vector rom xyz
calculate basis vectors for normal
sampleCount = 0
irradiance = 0
for phi 0..2π with increment 0.025 do:
    for theta in 0..π/2 with increment 0.025 do:
        sampleVector = fromSphericalToCartesian(phi, theta)
        transform sampleVector into the basis of normal
        irradianceSample = texture(environmentMap, sampleVector)
        irradianceSample *= cos(theta) * sin(theta)
        irradiance += irradianceSample
        sampleCount++

irradiance = π * irradiance / sampleCount
store(irradianceMap, xyz, irradiance)
```

Výpis kódu 6: Pseudokód algoritmu pro generování Irradiance Mapy, založeno na kódu z LearnOpenGL [46].

V tomto algoritmu se hemisféra vzorkuje po diskrétních krocích `phi` a `theta`. Vždy je potřeba `phi` a `theta` přepočítat do kartézských souřadnic (`sampleVector`) a poté převést do báze normály.

Výpočet DFG složky a LD složky pro spekulární část IBL se nese ve stejném duchu, ale pro jejich výpočet je použito vzorkování podle důležitosti. Pro výpočet DFG složky byl použit kód z Real Shading in Unreal Engine 4 [28] převedený do podoby compute shaderu. Zde je pro stručnost uveden pouze pseudokód:

```
load xy coordinates from gl_GlobalInvocationID
calculate LUTCoordinates (representing (n . v) and roughness) from xy
dfg1 = 0
dfg2 = 0
sampleCount = 0
n = vec3(0, 0, 1); // Libovolný normálový vektor
vec3 v;            // Nastavení vektoru v tak, aby sedělo (n . v)
v.x = sqrt(1. - (n . v) * (n . v))
v.y = 0;
v.z = (n . v)

for i in 0..SAMPLE_COUNT do:
    vec2 Xi = hammersley(i, SAMPLE_COUNT)
    vec3 H = importanceSampleGgx(Xi, N, roughness)
    vec3 L = normalize(2.0 * dot(V, H) * H - V)

    calculate dfg1 and dfg2 for current sample vector and add to total

store(dfgLUT, xy, vec2(dfg1, dfg2) / SAMPLE_COUNT)
```

Výpis kódu 7: Pseudokód pro výpočet DFG textury, založeno na kódu z Real Shading in Unreal Engine 4 [28].

Nejdůležitější částí algoritmu je `for loop`, ve kterém probíhá Monte Carlo integrace. Jak již bylo zmíněno, klíčové je použití vzorkování podle důležitosti. Vygenerování preferovaného vektoru **l** je rozděleno na 3 kroky. Prvně je použita funkce Hammersley, která mapuje vzorky od 0 do N na body $\mathbf{X_i}$ v jednotkovém čtverci. Ty je poté možné jednoduše transformovat na vzorkové vektory v jednotkové hemisféře [48]. V případě použitého algoritmu je ale bod $\mathbf{X_i}$ využit ve funkci `importanceSampleGgx`, která implementuje vzorkování podle důležitosti pro GGX distribuci. Funkce `importanceSampleGgx` tedy podle bodu $\mathbf{X_i}$, normály a drsnosti vygeneruje preferovaný halfway vektor. Ve třetím kroku se pomocí halfway vektoru spočítá vektor **l**. Nakonec se už jen podle rovnic 21 a 22 vypočítají části složek DFG_1 a DFG_2.

Compute shader pro výpočet LD cubemapy je invokován podobně jako shader pro výpočet irradiance mapy, tzn. **xy** souřadnice `gl_GlobalInvocationID` určují souřadnice pixelu ve stěně cubemapy a souřadnice **z** určuje stěnu cubemapy. Shader je invokován jednou pro každou úroveň mipmapy. Při každé invokaci je shaderu nastavena jiná hodnota drsnosti pomocí `uniform float` parametru. Algoritmus pro integraci LD části rovnice 20 je podobný algoritmu pro DFG část:

```
load xyz coordinates from gl_GlobalInvocationID
calculate normal vector rom xyz
prefilteredColor = 0
totalWeight = 0
for i in 0..SAMPLE_COUNT do:
    vec2 Xi = hammersley(i, SAMPLE_COUNT)
    vec3 H = importanceSampleGgx(Xi, N, roughness)
    vec3 L = normalize(2.0 * dot(V, H) * H - V)

    NoL = dot(N, L)
    if (NoL > 0):
        prefilteredColor += texture(environmemntMap, L);
        totalWeight += NoL;

store(ldTexture, xyz, prefilteredColor / totalWeight)
```

Výpis kódu 8: Pseudokód pro výpočet LD části, převzato z Real Shading in Unreal Engine 4 [28].

Sébastien Lagarde z Electronic Arts navrhuje vylepšit tento algoritmus pomocí techniky „před-filtrovaného vzorkování podle důležitosti“ [18], která je popsána v knize Gpu Gems 3 [49]. Základní idea této techniky je, že podle toho, jakou má daný vzorkový vektor pravděpodobnost, je vhodné připočítat osvětlení z různě širokých

prostorových úhlů okolo vzorkového vektoru. Pro vzorkové vektory s nízkou pravděpodobností je vhodné připočítat osvětlení z širšího prostorového úhlu, kdežto pro vzorkové vektory s vysokou pravděpodobností z užšího prostorového úhlu. Toto lze jednoduše implementovat pomocí vzorkování mip-úrovní mapy prostředí. Hlavním důvodem pro použití této techniky je snížení množství vzorků, které je nutné spočítat.

7.5.2.3 Kvalita předzpracování

Při testování implementace předzpracování bylo řešeno několik problémů. Nejobtížnější problém se týkal kvality předvypočítaných textur, ta totiž závisí na dvou faktorech: počtu vzorků při integraci a typu environment mapy. Pro metody Monte Carlo platí, že čím je počet vzorků vyšší, tím je odhad integrálu lepší. Vhodné je však použít co nejmenší počet vzorků tak, aby kvalita obrazu byla dostatečná.

Při testování vlastní implementace bylo zjištěno, že typ environment mapy může mít silně negativní vliv na množství vzorků potřebného k dosažení dobré kvality obrazu. Typickým problémovým případem jsou například environment mapy z venkovního prostředí, které obsahují slunce. Slunce vytváří extrémní množství záře, které je ale koncentrováno pouze do několika sousedních pixelů. Pro tyto environment mapy je nutné podstatně zvýšit množství vzorků, jinak předvypočítané textury obsahují artefakty. Nevýhodou navýšení množství vzorků je zároveň masivní zvýšení času potřebného pro předvýpočet textur. Například pro zmíněné problematické environment mapy může předvýpočet trvat více než minutu.

7.5.3 Osvětlení pomocí naměřených BRDF

V kapitole 3.1.1 byla zmíněna možnost využití naměřených BRDF přímo v rendereru. Tato reprezentace BRDF je snadno využitelná pro osvětlení bodovými zdroji světla. Naopak není možné smysluplné využití techniky IBL v reálném čase.

Osvětlení pomocí těchto BRDF bylo implementováno ve fragment shaderu `data_driven.frag`. Jádro shaderu tvoří algoritmus typický pro výpočet osvětlení bodovými zdroji světla:

```
vec3 totalLight = vec3(0.);
for lightSource in lightSources:
    vec3 light = lightSource.color;
    vec3 lightDirection = normalize(lightSource.pos - fragmentPos);
    float NoL = dot(normal, lightDirection);

    vec3 brdf = lookupMeasuredBrdf(...);
    totalLight += light * brdf * NoL;

return totalLight;
```

Výpis kódu 9: Algoritmus pro osvětlení bodovými zdroji světla pomocí naměřených BRDF.

Klíčová je funkce lookupMeasuredBrdf, která pomocí parametrů směru osvětlení, směru pozorovatele, normály, tangenty a bitangenty vyhodnotí danou BRDF. V této práci jsou použity 2 databáze BRDF: MERL databáze [8] a UTIA databáze [10]. MERL databáze obsahuje izotropní materiály a UTIA databáze převážně anizotropní materiály. Každá databáze typicky používá rozdílný formát pro uložení dat, a proto také probíhá načtení dat z databáze rozdílným způsobem.

K oběma databázím je přiložen zdrojový kód v jazyce C++, pomocí kterého lze data načíst. Tento kód byl přepsán do GLSL a lze jej tedy využít ve fragment shaderu. V obou případech bylo potřeba přepočítat vektor osvětlení, vektor pohledu, normálu, tangentu a bitangentu na úhly ϕ_i, ϕ_o, θ_i a θ_o, viz kapitola 3.

Naměřené BRDF materiály většinou zabírají docela významné množství paměti. Konkrétně MERL materiály vyžadují cca 30 MB a UTIA materiály cca 2 MB. V obou případech se jedná o pole 64-bitových čísel s plovoucí řádovou čárkou. Pro zpřístupnění takového objemu dat fragment shaderu byly využity Shader Storage Buffer Objects (SSBO), které byly přidány do OpenGL ve verzi 4.3.

Vytvoření SSBO a naplnění daty vypadá velice podobně jako vytvoření klasického OpenGL bufferu. Naopak deklarace bufferu v samotném shaderu se liší, SSBO se v shaderu deklaruje pomocí následující syntaxe:

```
layout(std430, binding = ...) buffer jmenoBufferu { obsah bufferu ... };
```

7.5.4 Tone-mapping

Výstupem fyzikálně založeného rendereru je zář, anglicky „radiance“, jejíž hodnoty se nacházejí v intervalu od nuly do nekonečna. To znamená, že mají vysoký dynamický rozsah (HDR). Před výstupem na monitor je nutné převést záři na RGB

hodnoty v nízkém dynamickém rozsahu (LDR) od nuly do jedné. Proces konverze hodnot z HDR do LDR se nazývá tone-mapping. Cílem tone-mappingu je převedení hodnot z HDR do LDR tak, aby se zachovala vysoká kvalita obrazu. V navrženém rendereru byly implementovány dva různé tzv. „tone-mapping operátory“: jednoduchý „Reinhard operátor“ [50] a operátor z počítačové hry Uncharted 2 [51].

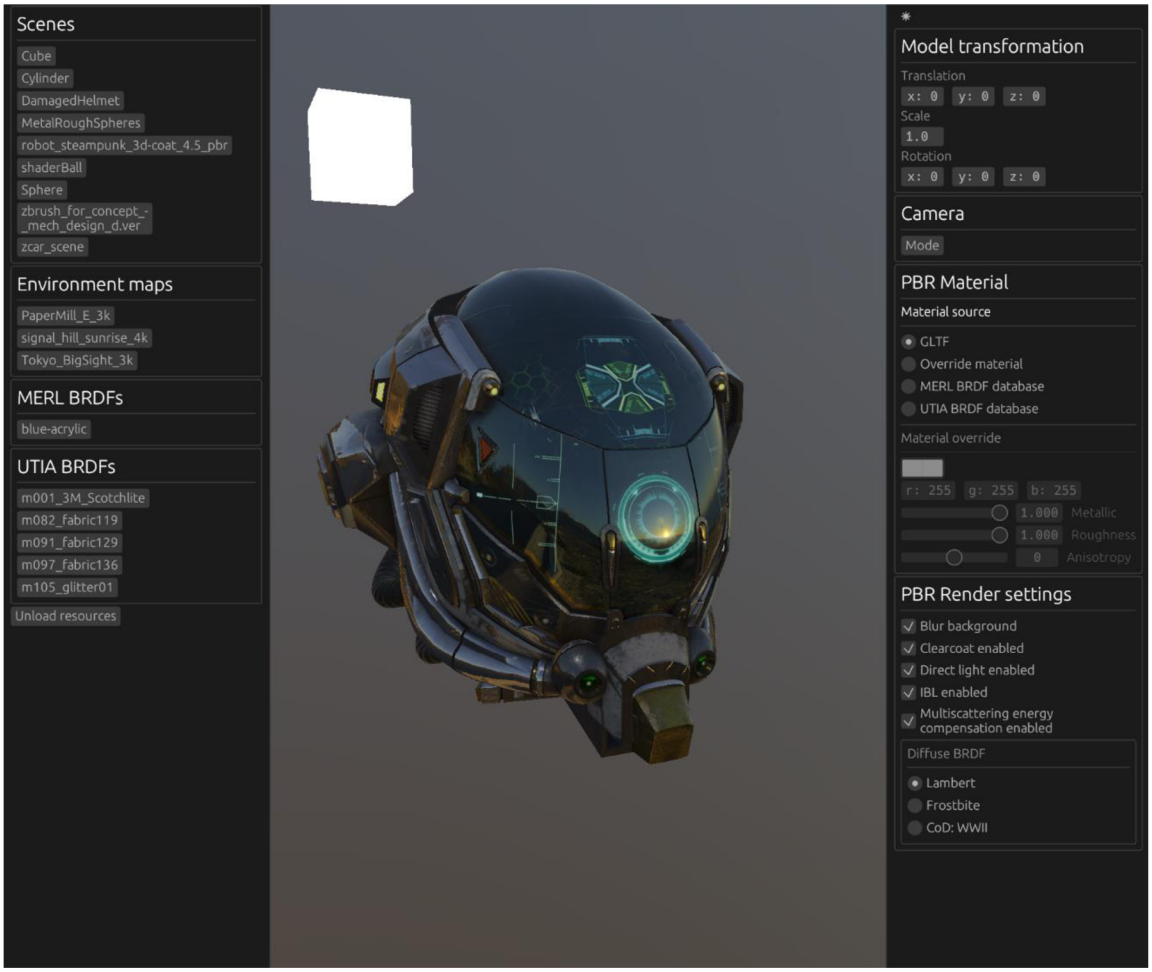
8 Shrnutí výsledků

V rámci práce byla vytvořena aplikace testující popsané zobrazovací metody. Zdrojový kód a binární verze aplikace jsou dostupné na platformě GitHub[5]. Pro použití BRDF z databází MERL [8] nebo UTIA [10] je nutné soubory s BRDF stáhnout z webů autorů a umístit je do složek `resources/BRDFDatabase/brdfs`, respektive `resources/UTIA`.

GUI aplikace lze vidět na Obr. 20. V levém panelu jsou výpisy jednotlivých zdrojů, tj. GLTF scén, environment map, MERL BRDF a UTIA BRDF. Kliknutím na jednotlivé názvy se daný zdroj zvolí a bude použit pro vykreslování. Načtení zdroje může trvat několik sekund. Pro vykreslení pomocí MERL nebo UTIA BRDF je nutné také v pravém panelu zvolit příslušný materiál.

V pravém panelu jsou umístěny elementy k ovládání vykreslování. V sekci „Model transformation“ lze ovládat pozici, velikost a rotaci zobrazovaného objektu. V následujícím oddílu lze zvolit typ kamery. Kamera se ovládá pomocí kláves WSAD a pravého tlačítka myši. V části „PBR Material“ lze určit, jakým způsobem bude nastaven materiál zobrazovaného objektu. Základní nastavení „GLTF“ načítá parametry z GLTF souboru. Varianta „Override material“ umožňuje nastavit jednotlivé parametry modelu Metallic-Roughness a anizotropie. Tato varianta funguje nejlépe s modely, které nemají textury, jako jsou „Sphere“, „Cube“ nebo „Cylinder“. Možnosti „MERL BRDF database“ a „UTIA BRDF database“ umožňují zvolit jako zdroj materiálu jednotlivé materiály z daných databází. V následující sekci „PBR Render settings“ lze vypnout a zapnout různé vlastnosti vykreslování.

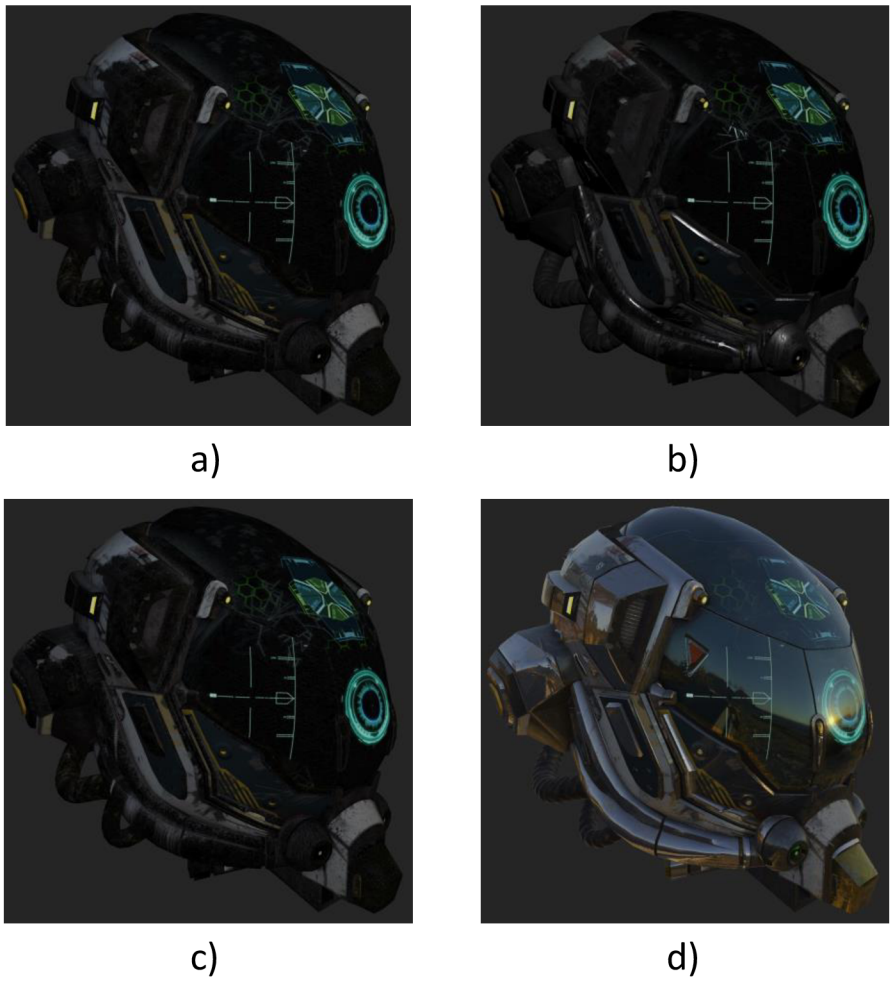
[5] Zdrojový kód: https://github.com/TomasKralCZ/PBR.
Binární verze: https://github.com/TomasKralCZ/PBR/releases.



Obr. 20 GUI vytvořené aplikace.
Vlastní tvorba.

8.1 Testování zobrazovacích metod

Pro testování hlavního PBR shaderu byl použit model „Battle Damaged Sci-fi Helmet - PBR“ [52] od tvůrce theblueturtle_ [53], zveřejněné pod licencí CC BY-NC 3.0 [54].



Obr. 21 Zobrazení modelu „Battle Damaged Sci-fi Helmet – PBR“ pomocí různých metod: a) bodový zdroj světla pouze s difusní složkou, b) bodový zdroj světla s difusní i spekulární složkou, c) IBL pouze s difusní složkou, d) IBL s difusní i spekulární složkou.
Vlastní tvorba.

Dále byl porovnán BRDF model s jednorázovým odrazem oproti modelu s vícenásobným odrazem, viz kapitola 4.1.3. Srovnání lze vidět na Obr. 22. V této scéně lze vidět 5 koulí s proměnlivou drsností povrchu – drsnost se zleva doprava zvyšuje od 0 do 1. Všechny koule mají nastavenou kovovost na 1 barvu na RGB(255, 255, 255). Všimněte si příliš temného povrchu drsné koule v případě modelu s jednorázovým odrazem. Tento rozdíl je ještě více markantní v tzv. „white furnace testu“, což je test kdy se osvětlení z okolí nastaví na konstantu. V takovém případě by měly být kompletně kovové materiály se 100% odrazivostí neviditelné. V případě jednorázového odrazu však dochází ke ztrátě energie, viz Obr. 23.

a)

b)

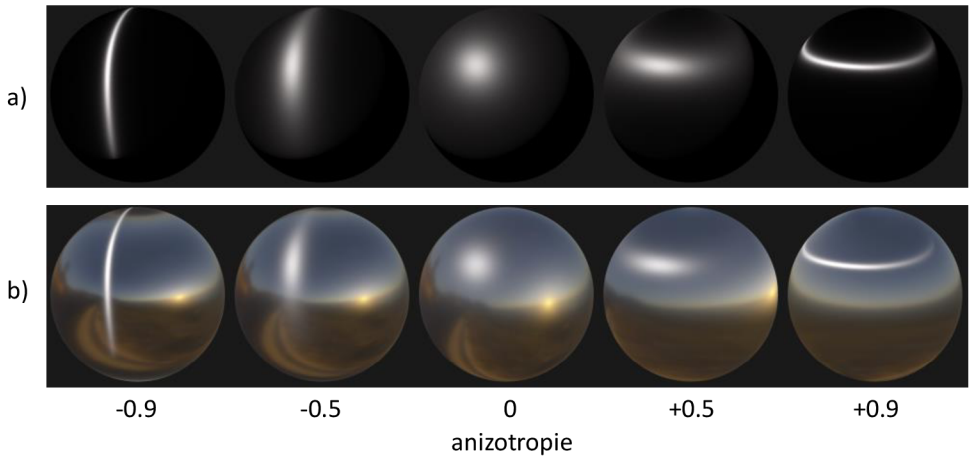
Obr. 22 Srovnání modelu s jednorázovým odrazem (a) a modelu s vícenásobným odrazem (b).
Vlastní tvorba.

a)

b)

Obr. 23 Srovnání modelu s jednorázovým odrazem (a) a modelu s vícenásobným odrazem (b).
Vlastní tvorba.

Na Obr. 24 můžete vidět vykreslení materiálů s různou úrovní anizotropie. Všechny koule mají nastavenou drsnost na 0.5. Kovovost je nastavena na 1 a barva na RGB(255, 255, 255).

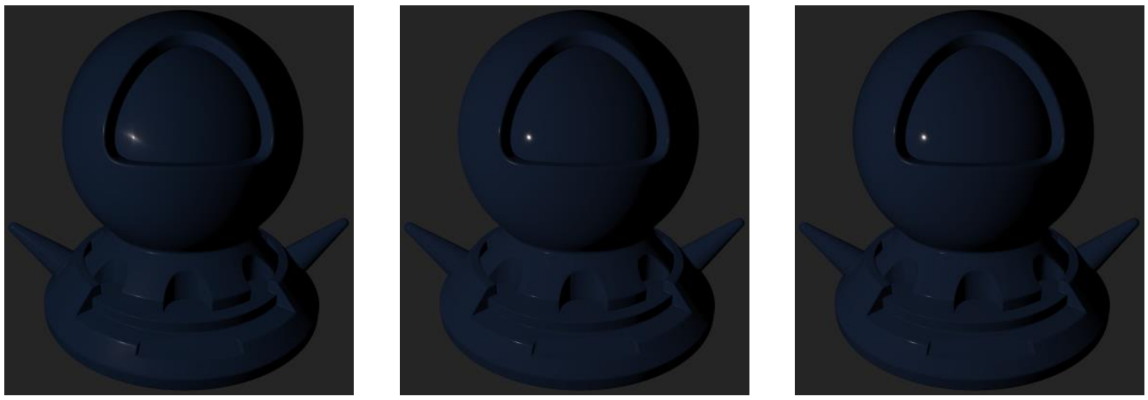


Obr. 24 Vykreslení anizotropního materiálu s různými osvětlovacími metodami: a) bodový zdroj světla, b) IBL společně s bodovým zdrojem světla. Vlastní tvorba.

8.1.1 Srovnání analytických BRDF modelů s naměřenými BRDF

Pro toto srovnání byly zvoleny naměřené BRDF z MERL databáze. Ty byly porovnány s analytickým BRDF modelem skládající se ze spekulární časti (mikroploškový model viz. kapitola 7.5.) a 2 difusních modelů: Lambert a „Call of Duty: WWII“ difusní model.

První materiál, který byl testován je „blue-acrylic“ neboli modrá akrylová barva. Výsledky lze vidět na Obr. 25. Pro testování byl použit 3D model „Shader Ball“ [55] od tvůrce athirn [56], zveřejněný pod licencí CC-BY-4.0 [57]. Parametry analytického modelu byly nastaveny na: drsnost 0.66, kovovost: 0, barva: RGB(4, 10, 29). Parametr drsnosti byl zvolen tak, aby byl co nejvíce podobný tvar spekulárních odlesků, nikoliv vzhled difusní složky.



Obr. 25 Porovnání MERL materiálu blue-acrylic s dvěma difusními modely. Vlastní tvorba.

Rozdíl mezi difusními modely je malý, ale pozorovatelný. Všimněte si barvy okraje horní části objektu. U MERL materiálu je okraj mírně ztmavený, což poměrně dobře replikuje Call of Duty: WWII model. Naopak u modelu Lambert je okraj příliš světlý.

Druhý testovaný materiál byl „red-plastic“, neboli červený plast. Výsledky jsou vidět na Obr. 26. Parametry analytického modelu byly nastaveny na: drsnost: 0.6, kovovost: 0, barva: RGB(68, 10, 0). Jak můžete vidět na levé horní části objektu, MERL materiál vykazuje vysokou míru retroreflexe, která je charakteristická pro drsné materiály. Call of Duty: WWII model retroreflexi vykazuje, ale zdaleka ne tak silně, aby to odpovídalo naměřenému materiálu. Naopak Lambert model s retroreflexí vůbec nepočítá, a proto u něj dochází na okraji objektu k přílišnému ztmavení.

Obr. 26 Porovnání MERL materiálu red-plastic s dvěma difusními modely. Vlastní tvorba.

9 Závěry a doporučení

V bakalářské práci byly prozkoumány zobrazovací metody založené na fyzikálních modelech – physically based rendering methods (PBR). V teoretické části byly nejprve popsány základní fyzikální zákony popisující odraz světla. Následně bylo věnováno mnoho prostoru fyzikálně založeným dvousměrovým distribučním funkcím odrazu světla (BRDF) založeným na principu mikroplošek. Dále byly popsány běžné osvětlovací techniky jako jsou bodové zdroje světla a osvětlení na základě obrazu.

V praktické části byla navržena a implementována aplikace, která umožňuje vykreslit GLTF scény s využitím zobrazovacích metod založených na PBR. Mezi implementované PBR techniky patří jedna mikroplošková spekulární BRDF a dvě difusní BRDF. Dále bylo implementováno osvětlení na základě obrazu, dodávající výslednému obrazu značnou pestrost. V neposlední řadě bylo implementováno vykreslení scény pomocí naměřených BRDF z dvou různých materiálových databází.

V textu praktické části byl nastíněn design rendereru a způsob načítání GLTF souborů. Dále byl popsán způsob, kterým aplikace pracuje s GLSL shadery. Nejobsáhlejší část textu se zabývala implementací osvětlovacích algoritmů pomocí programovatelných grafických karet. Práce s OpenGL a se shaderovým jazykem GLSL se ukázala být náročná. Je zřejmé, že nástroje pro práci s GLSL nejsou zdaleka tak rozvinuté, jako nástroje pro běžné programovací jazyky.

V závěru práce bylo představeno výsledné grafické rozhraní aplikace a následně byly otestovány zobrazovací metody. Byl porovnán mikroploškový model s jednorázovým odrazem světla a model s vícenásobným odrazem světla. Z porovnání je zřejmé, že model s vícenásobným odrazem světla je matematicky přesnější a vizuálně kvalitnější. Dále byly porovnány difusní BRDF modely s naměřenými materiály z MERL databáze. Z porovnání vychází najevo, že BRDF model ze hry Call of Duty: WWII je přesnější než klasický Lambertovský difusní model, za cenu vyšších výpočetních nároků. Avšak vizuální rozdíly mezi těmito modely jsou v některých případech sotva patrné. Rozhodnutí, zda použít rychlejší Lambertovský model nebo přesnější „Call of Duty: WWII“ model tak závisí na potřebách konkrétní aplikace.

Práci by bylo možné rozšířit několika způsoby. Například kapitola o mikroploškových modelech byla cílena na grafické programátory, a proto nebyly v textu do detailu probrány všechny matematické vlastnosti mikroploškových modelů. Tuto problematiku výborně vysvětluje článek „Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs“ [21]. Dále by bylo vhodné implementovat další modely osvětlení. Často používaná technika v této oblasti je například „screen-space reflections“ [58], případně se nabízí plošné zdroje světla založené na technice „linearly-transformed cosines“ [32]. V neposlední řadě by bylo zajímavé implementovat BRDF pro specifické materiály, jako jsou například iridescentní materiály.

10 Seznam použité literatury

[1] MCDERMOTT, Wes. *The PBR Guide: A Handbook for Physically Based Rendering* [online]. 3. vyd. Clermont-Ferrand: Allegorithmic, 2018 [vid. 2022-06-18]. ISBN 978-2-490-07100-5. Dostupné z: https://substance3d.adobe.com/tutorials/courses/the-pbr-guide-part-2

[2] HANIKA, Johannes. Manuka: Weta Digital's spectral renderer. In: *ACM SIGGRAPH 2017 Courses*: *Path Tracing in Production - Part 1: Production Renderers* [online]. 2017, s. 13:1-13:39. SIGGRAPH '17. ISBN 978-1-4503-5014-3. Dostupné z: doi:10.1145/3084873.3084904

[3] NIMIER-DAVID, Merlin, Delio VICINI, Tizian ZELTNER a Wenzel JAKOB. Mitsuba 2: A Retargetable Forward and Inverse Renderer. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* [online]. 2019, **38**(6). Dostupné z: doi:10.1145/3355089.3356498

[4] AKENINE-MÖLLER, Tomas, Eric HAINES, Naty HOFFMAN, Angelo PESCE, Michał IWANICKI a Sébastien HILLAIRE. *Real-Time Rendering*. 4. vyd. Boca Raton, FL, USA: A K Peters/CRC Press, 2018. ISBN 978-1-138-62700-0.

[5] JIMENEZ, Jorge, Károly ZSOLNAI, Adrian JARABO, Christian FREUDE, Thomas AUZINGER, Xian-Chun WU, Javier DER PAHLEN, Michael WIMMER a Diego GUTIERREZ. Separable Subsurface Scattering. *Computer Graphics Forum* [online]. 2015, **34**(6), 188–197. ISSN 0167-7055. Dostupné z: doi:10.1111/cgf.12529

[6] WANG, Lifeng, Wenle WANG, Julie DORSEY, Xu YANG, Baining GUO a Heung-Yeung SHUM. Real-time rendering of plant leaves. In: *ACM SIGGRAPH 2006 Courses* [online]. New York, NY, USA: Association for Computing Machinery, 2006, s. 5-es [vid. 2022-06-22]. ISBN 978-1-59593-364-5. Dostupné z: doi:10.1145/1185657.1185725

[7] MATT PHARR, JAKOB WENZEL, a HUMPHREYS GREG. *Physically Based Rendering: From Theory To Implementation* [online]. 5. červen 2022 [vid. 2022-06-05]. Dostupné z: https://pbr-book.org/3ed-2018/Introduction#

[8] MATUSIK, Wojciech, Hanspeter PFISTER, Matt BRAND a Leonard MCMILLAN. A Data-Driven Reflectance Model. *ACM Transactions on Graphics* [online]. 2003, **22**(3), 759–769. Dostupné z: doi:https://doi.org/10.1145/882262.882343

[9] LAGARDE, Sébastien. Physically-Based Materials: Where Are We? In: *SIGGRAPH '17: ACM SIGGRAPH 2017 Courses*: *Open problems in real-time rendering* [online]. New York, NY, USA: Association for Computing Machinery, 2017. ISBN 978-1-4503-5014-3. Dostupné z: https://openproblems.realtimerendering.com/s2017/index.html

[10] FILIP, J. a R. VÁVRA. Template-Based Sampling of Anisotropic BRDFs. *Computer Graphics Forum* [online]. 2014, **33**(7), 91–99. ISSN 01677055. Dostupné z: doi:https://doi.org/10.1111/cgf.12477

[11] SOLDADO, Rosana Montes a Carlos Ureña ALMAGRO. *An Overview of BRDF Models*. B.m.: University of Granada, Granada, Spain. 2012.

[12] NGAN, Addy, Frédo DURAND a Wojciech MATUSIK. Experimental Analysis of BRDF Models. In: *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques* [online]. Goslar: Eurographics Association, 2005, s. 117–126. ISBN 3-905673-23-1. Dostupné z: doi:10.2312/EGWR/EGSR05/117-126

[13] COOK, R. L. a K. E. TORRANCE. A Reflectance Model for Computer Graphics. *ACM Transactions on Graphics* [online]. 1982, **1**(1), 7–24. ISSN 0730-0301. Dostupné z: doi:10.1145/357290.357293

[14] GUY, Romain a Mathias AGOPIAN. Physically Based Rendering in Filament. *Physically Based Rendering in Filament* [online]. 20. únor 2019 [vid. 2022-07-07]. Dostupné z: https://google.github.io/filament/Filament.md.html

[15] OREN, Michael a Shree K. NAYAR. Generalization of Lambert's reflectance model. In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* [online]. New York, NY, USA: Association for Computing Machinery, 1994, s. 239–246 [vid. 2022-06-27]. ISBN 978-0-89791-667-7. Dostupné z: doi:10.1145/192161.192213

[16] TROWBRIDGE, T. S. a K. P. REITZ. Average irregularity representation of a rough surface for ray reflection. *J. Opt. Soc. Am.* [online]. 1975, **65**(5), 531–536. Dostupné z: doi:10.1364/JOSA.65.000531

[17] MCAULEY, Stephen. Rendering the World of Far Cry 4. In: [online]. B.m. 2018 [vid. 2022-11-14]. Dostupné z: https://www.youtube.com/watch?v=rD6KcxcCl_8

[18] LAGARDE, Sébastian a Charles DE ROUSIERS. Moving Frostbite to PBR. In: *ACM SIGGRAPH 2014 Courses*: *Physically Based Shading in Theory and Practice* [online]. New York, NY, USA: Association for Computing Machinery, nedatováno. ISBN 978-1-4503-2962-0. Dostupné z: doi:10.1145/2614028.2615431

[19] KULLA, Christopher a Alejandro CONTY. Revisiting Physically Based Shading at Imageworks. In: *ACM SIGGRAPH 2017 Courses*: *Physically Based Shading in Theory and Practice* [online]. New York, NY, USA: Association for Computing Machinery, 2017, s. 136. ISBN 978-1-4503-5014-3. Dostupné z: doi:10.1145/3084873.3084893

[20] WALTER, Bruce, Stephen MARSCHNER, Hongsong LI a Kenneth TORRANCE. Microfacet Models for Refraction through Rough Surfaces. In: *Eurographics*

Symposium on Rendering: *Rendering Techniques* [online]. Goslar: The Eurographics Association, 2007, s. 195–206. ISBN 978-3-905673-52-4. Dostupné z: doi:10.2312/EGWR/EGSR07/195-206

[21] HEITZ, Eric. Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs. 2014, **3**(2), 61.

[22] SMITH, B. Geometrical shadowing of a random rough surface. *IEEE Transactions on Antennas and Propagation* [online]. 1967, **15**(5), 668–671. ISSN 1558-2221. Dostupné z: doi:10.1109/TAP.1967.1138991

[23] KARIS, Brian. Graphic Rants: Specular BRDF Reference. *Graphic Rants* [online]. 3. srpen 2013 [vid. 2022-06-27]. Dostupné z: http://graphicrants.blogspot.com/2013/08/specular-brdf-reference.html

[24] FDEZ-AGÜERA, Carmelo J. A Multiple-Scattering Microfacet Model for Real-Time Image Based Lighting. *Journal of Computer Graphics Techniques (JCGT)*. 2019, **8**(1), 45–55. ISSN 2331-7418.

[25] SCHLICK, Christophe. An Inexpensive BRDF Model for Physically-based Rendering. *Computer Graphics Forum* [online]. 1994, **13**(3), 233–246. ISSN 1467-8659. Dostupné z: doi:10.1111/1467-8659.1330233

[26] HOFFMAN, Naty. Some Thoughts on the Fresnel Term. In: *ACM SIGGRAPH 2020 Courses*: *Physically Based Shading in Theory and Practice* [online]. Virtual Event, USA: Association for Computing Machinery, 2020. ISBN 978-1-4503-7972-4. Dostupné z: doi:https://doi.org/10.1145/3388769.3407523

[27] BELCOUR, Laurent, Mégane BATI a Pascal BARLA. Bringing an Accurate Fresnel to Real-Time Rendering: a Preintegrable Decomposition. In: *ACM SIGGRAPH 2020 Courses*: *Physically Based Shading in Theory and Practice* [online]. Virtual Event, USA: Association for Computing Machinery, 2020. ISBN 978-1-4503-7972-4. Dostupné z: doi:https://doi.org/10.1145/3388769.3407523

[28] KARIS, Brian. Real Shading in Unreal Engine 4. In: *ACM SIGGRAPH 2013 Courses*: *Physically based shading in theory and practice* [online]. New York, NY, USA: Association for Computing Machinery, 2013. ISBN 978-1-4503-2339-0. Dostupné z: doi:10.1145/2504435.2504457

[29] BURLEY, Brent. Physically-Based Shading at Disney. In: *ACM SIGGRAPH 2012 Courses*: *Practical Physically-Based Shading in Film and Game Production* [online]. New York, NY, USA: Association for Computing Machinery, 2012. ISBN 978-1-4503-1678-1. Dostupné z: doi:10.1145/2343483.2343493

[30] CHAN, Danny. Material Advances in Call of Duty: WWII. In: *ACM SIGGRAPH 2018 Courses*: *Advances in Real-Time Rendering in Games: Part I* [online]. Vancouver British Columbia Canada: Association for Computing Machinery,

2018. SIGGRAPH '18. ISBN 978-1-4503-5809-5. Dostupné z: doi:10.1145/3214834.3264541

[31] BELCOUR, Laurent. Efficient Rendering of Layered Materials using an Atomic Decomposition with Statistical Operators. *ACM Transactions on Graphics* [online]. 2018, **37**(4), 1. Dostupné z: doi:10.1145/3197517.3201289

[32] HEITZ, Eric, Jonathan DUPUY, Stephen HILL a David NEUBELT. Real-time polygonal-light shading with linearly transformed cosines. *ACM Transactions on Graphics* [online]. 2016, **35**(4), 41:1-41:8. ISSN 0730-0301. Dostupné z: doi:10.1145/2897824.2925895

[33] KT, Aakash, Eric HEITZ, Jonathan DUPUY a P. J. NARAYANAN. Bringing Linearly Transformed Cosines to Anisotropic GGX. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* [online]. 2022, **5**(1), 1–18. ISSN 2577-6193. Dostupné z: doi:10.1145/3522612

[34] RAMAMOORTHI, Ravi a Pat HANRAHAN. An Efficient Representation for Irradiance Environment Maps. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2001)*. 2001, **20**(3), 497–500.

[35] THE KHRONOS® 3D FORMATS WORKING GROUP. *glTF™ 2.0 Specification* [online]. 10. listopad 2021 [vid. 2022-07-06]. Dostupné z: https://www.khronos.org/registry/glTF/specs/2.0/glTF-2.0.pdf

[36] THE KHRONOS® 3D FORMATS WORKING GROUP. glTF - Runtime 3D Asset Delivery. *The Khronos Group* [online]. 3. prosinec 2020 [vid. 2023-02-09]. Dostupné z: https://www.khronos.org/gltf/

[37] THE KHRONOS® 3D FORMATS WORKING GROUP. KHR_materials_clearcoat. *KHR_materials_clearcoat* [online]. 3. listopad 2022 [vid. 2022-11-03]. Dostupné z: https://github.com/KhronosGroup/glTF/blob/2a9996a2ea66ab712590eaf62f39f1115996f5a3/extensions/2.0/Khronos/KHR_materials_clearcoat/README.md

[38] THE KHRONOS® 3D FORMATS WORKING GROUP. KHR_materials_anisotropy extension branch. *GitHub* [online]. 2. říjen 2023 [vid. 2023-02-10]. Dostupné z: https://github.com/KhronosGroup/glTF/pull/1798

[39] ERNERFELDT, Emil. *egui: an easy-to-use GUI in pure Rust* [online]. Rust. 10. duben 2023 [vid. 2023-04-10]. Dostupné z: https://github.com/emilk/egui

[40] CORNUT, Omar. *Dear ImGui* [online]. C++. 10. duben 2023 [vid. 2023-04-10]. Dostupné z: https://github.com/ocornut/imgui

[41] LANTINGA, Sam. *Simple DirectMedia Layer - Homepage* [online]. [vid. 2023-04-10]. Dostupné z: https://www.libsdl.org/

[42] MIKKELSEN, Morten. MikkTSpace. *MikkTSpace* [online]. [vid. 2023-03-02]. Dostupné z: http://www.mikktspace.com/

[43] LENGYEL, Eric. *Foundations of game engine development 2*. Lincoln, California: Terathon Software LLC, 2019. Foundations of game engine development. ISBN 978-0-9858117-5-4.

[44] ANONYMOUS. *An OpenGL preprocessor for Rust – CodeCrash* [online]. 27. září 2020 [vid. 2023-03-02]. Dostupné z: https://codecrash.me/an-opengl-preprocessor-for-rust

[45] PROUILLET, Vincent. *Tera* [online]. [vid. 2023-03-02]. Dostupné z: https://tera.netlify.app/

[46] DE VRIES, Joey. LearnOpenGL - Diffuse irradiance. *Learn OpenGL* [online]. [vid. 2023-02-17]. Dostupné z: https://learnopengl.com/PBR/IBL/Diffuse-irradiance

[47] IBLBaker. *The Kreature Experiment* [online]. [vid. 2023-02-21]. Dostupné z: http://www.derkreature.com/iblbaker

[48] DAMMERTZ, Holger. *Points on a Hemisphere* [online]. 7. listopad 2012 [vid. 2023-01-20]. Dostupné z: http://holger.dammertz.org/stuff/notes_HammersleyOnHemisphere.html

[49] COLBERT, Mark a Jaroslav KŘIVÁNEK. GPU-based Importance Sampling. In: *GPU Gems 3* [online]. B.m.: Addison-Wesley Professional, 2007. ISBN 0-321-51526-9. Dostupné z: https://developer.nvidia.com/gpugems/gpugems3/part-iii-rendering/chapter-20-gpu-based-importance-sampling

[50] REINHARD, Erik, Michael STARK, Peter SHIRLEY a James FERWERDA. Photographic tone reproduction for digital images. *ACM Transactions on Graphics* [online]. 2002, **21**(3), 267–276. ISSN 0730-0301. Dostupné z: doi:10.1145/566654.566575

[51] HABLE, John. Uncharted 2: HDR Lighting. In: *GDC 2010* [online]. B.m. [vid. 2023-02-28]. Dostupné z: https://gdcvault.com/play/1012351/Uncharted-2-HDR

[52] THEBLUETURTLE_. *Battle Damaged Sci-fi Helmet - PBR - 3D model by theblueturtle_ (@theblueturtle_) [b81008d]* [online]. [vid. 2023-04-10]. Dostupné z: https://sketchfab.com/3d-models/battle-damaged-sci-fi-helmet-pbr-b81008d513954189a063ff901f7abfe4

[53] THEBLUETURTLE_. theblueturtle_. *Sketchfab* [online]. [vid. 2023-04-10]. Dostupné z: https://sketchfab.com/theblueturtle_

[54] CREATIVE COMMONS. *Creative Commons — Attribution-NonCommercial 3.0 Unported — CC BY-NC 3.0* [online]. [vid. 2023-04-10]. Dostupné z: https://creativecommons.org/licenses/by-nc/3.0/

[55] ATHIRN. *Shader Ball - Download Free 3D model by athirn (@athirn) [68d28c0]* [online]. [vid. 2023-04-10]. Dostupné z: https://sketchfab.com/3d-models/shader-ball-68d28c0dd48745fea61fb4e7708217ea

[56] ATHIRN. *athirn - Sketchfab* [online]. [vid. 2023-04-10]. Dostupné z: https://sketchfab.com/athirn

[57] CREATIVE COMMONS. *Creative Commons — Attribution 4.0 International — CC BY 4.0* [online]. [vid. 2023-04-10]. Dostupné z: https://creativecommons.org/licenses/by/4.0/

[58] MCGUIRE, Morgan a Michael MARA. Efficient GPU Screen-Space Ray Tracing. *Journal of Computer Graphics Techniques (JCGT)*. 2014, **3**(4), 73–85. ISSN 2331-7418.

11 Přílohy

1) Elektronická příloha – ZIP archiv se zdrojovým kódem a spustitelnou formou aplikace.

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor: **Tomáš Král**

Studium: I2000379

Studijní program: B1802 Aplikovaná informatika

Studijní obor: Aplikovaná informatika

Název bakalářské práce: **Zobrazovací metody založené na fyzikálních modelech**

Název bakalářské práce AJ: Physically based rendering methods

Cíl, metody, literatura, předpoklady:

Cíl práce:

Cílem práce je prozkoumat přístupy pro zobrazení scény s využitím fyzikálně založených zobrazovacích metod – physically based rendering method (PBR). Pro vhodnou scénu navrhnout, implementovat a otestovat řešení pracující v reálném čase.

Postup prací:

1. Prozkoumat principy a metody PBR

2. Vytvořit přehled používaných přístupů a algoritmů pro výpočet osvětlení při vizualizaci prostorové scény s využitím PBR technik

3. Pro vybranou metodu navrhnout řešení s využitím programovatelných grafických karet

4. Navržené řešení implementovat a otestovat

5. Zhodnotit dosažené výsledky

[1] GUY, Romain a Mathias AGOPIAN. Physically Based Rendering in Filament. Physically Based Rendering in Filament [online]. 20. únor 2019 [vid. 2022-07-07]. Dostupné z: https://google.github.io/filament/Filament.md.html

[2] MCDERMOTT, Wes. The PBR Guide: A Handbook for Physically Based Rendering [online]. 3. vyd. Clermont-Ferrand: Allegorithmic, 2018 [vid. 2022-06-18]. ISBN 978-2-490-07100-5. Dostupné z: https://substance3d.adobe.com/tutorials/courses/the-pbr-guide-part-2

[3] BURLEY, Brent. Physically-Based Shading at Disney. In: ACM SIGGRAPH 2012 Courses: Practical Physically-Based Shading in Film and Game Production [online]. New York, NY, USA: Association for Computing Machinery, 2012. ISBN 978-1-4503-1678-1. Dostupné z: doi:10.1145/2343483.2343493

Zadávající pracoviště: Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu

Vedoucí práce: Ing. Bruno Ježek, Ph.D.

Datum zadání závěrečné práce: 26.1.2021