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

OSVĚTLENÍ PŘI VYKRESLOVÁNÍ VOLUMETRICKÝCH MODELŮ

Diplomová práce

Autor: Ondřej Caha
Studijní obor: Aplikovaná informatika

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

Hradec Králové listopad 2020

Prohlášení

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

V Zábřeze dne 16.11.2020 Ondřej Caha

Poděkování

Úvodem bych chtěl poděkovat vedoucímu této práce Ing. Bruno Ježkovi, Ph.D. za cenné rady a za vstřícný přístup při jejím zpracování.

Anotace

Osvětlení objemových dat je důležitým krokem k porozumění jejich obsahu a hloubkové informace, ale i v současné době je komplexní osvětlení výpočetně náročná záležitost. Často se jedná o kompromis mezi kvalitou výsledku a rychlostí vykreslování. Cílem práce je prozkoumat existující metody, porovnat jejich náročnost a zhodnotit dosažené výsledky. Jsou porovnávány dva přístupy na výpočet globálního osvětlení. První přístup počítá nový objem tak, aby každý jeho prvek reprezentoval své okolí. Pro tento objem jsou spočítány další úrovně mipmapy, kde prvek reprezentuje širší okolí. Toho je využito ve vykreslování pomocí trasování kuželem. Druhá metoda je autorská, která problém řeší předpočítáním stínů. Při změně pozice světla se světlo propaguje skrz celý objem plátek po plátku. Tento výpočet probíhá na grafické kartě s využitím compute shaderů. Výsledky ukazují, že oba tyto přístupy umožňují výpočet osvětlení v reálném čase. První z metod umožňuje lepší parametrizaci, ale autorská metoda dosahuje vyšší rychlosti vykreslování.

Annotation

Title: Lightning techniques for volume rendering

Lightning effects on volume data are necessary step for understanding of content and depth information of displayed data. Complex lightning is still computationally demanding task usually resulting in tradeoff between visual quality a rendering speed. This thesis aims for exploring existing methods, compare their computational complexity a evaluate results. There are two global illumination approaches compared in this thesis. The first approach computes new volume the way, that every element represents its vicinity. Mipmapping allows creating new volumes, where its elements represents wider area. Lightning is then computed by cone tracing. Second method, proposed by the author, is solving the problem by precomputing shadows, which are propagated through entire volume slice by slice from the light source. This is achieved using compute shader on graphic card. Results show both approaches are able to deliver global illumination effects in real time. First method allows better parameter control, but the author’s method provides higher rendering speed.

Obsah

1 Úvod

Na zobrazování objemových dat v reálném čase jsou v dnešní době již známé zavedené techniky umožňující zobrazit data jako medicínské skeny, výsledky simulací a stejně tak generovaná objemová data. Jejich zobrazení je však stále náročná záležitost. Vykreslování objemových dat oproti polygonálním modelům vyžaduje zobrazit data uvnitř objemu, pro polygonální data se zobrazí pouze povrch.

Běžně využívaná metoda ray casting je přímou implementací emisně-absorpčního optického modelu. Tento fyzikální model zahrnuje optické jevy absorpci a emisi světla, ale nezahrnuje jeho rozptyl, který výraznou měrou ovlivňuje finální obraz. Odraz světla a vržené stíny tak nejsou tímto modelem zachyceny, a přitom jsou tyto jevy důležitým krokem k porozumění obsahu vykreslovaného tělesa a hloubkové informace.

Pro dosažení chybějících efektů jsou aplikovány dva osvětlovací modely rozšiřující metodu ray castingu – lokální a globální osvětlovací model. Lokální osvětlovací model je typicky méně výpočetně náročný, avšak nedokáže zachytit komplexní osvětlení, například vržené stíny. Osvětlení je počítáno pouze z jeho okolí. Globální osvětlovací model zachycuje osvětlení komplexně, ale zvláště pro výpočet vícenásobného rozptylu je náročnost tohoto výpočtu v reálném čase příliš velká pro výpočetní výkon počítačů. Je třeba využívat zjednodušující předpoklady, předzpracovaná data nebo heuristické metody, aby bylo docíleno vykreslování v reálném čase.

V práci jsou porovnávány dvě metody zobrazující trojrozměrná data s globálním osvětlovacím modelem v reálném čase. Obě tyto metody leží na pomezí optických modelů jednoduchého rozptylu světla a vícenásobného rozptylu světla. Pro lepší vizuální vjem je spolu s těmito metodami aplikován lokální osvětlovací model a pro lepší výpočetní výkon jsou také aplikovány a testovány optimalizační metody.

Práce je rozdělena na dvě části. První – teoretická část popisuje fyzikální a teoretické základy, které jsou nutné pro zobrazení objemových dat, a některé standardní přístupy pro jejich zobrazení. Cílem práce bylo také vytvořit aplikaci zobrazující data s využitím osvětlovacích modelů. Druhá – praktická část obsahuje vlastní návrh implementace zkoumaných a navržených metod a zhodnocuje dosažené výsledky.

Kapitoly jsou uspořádány následovně. V kapitole číslo 2 jsou představeny fyzikální základy nutné pro vykreslování objemových dat. První jsou představeny optické jevy nastávající při průchodu světla materiálem, následují optické modely, které danou skutečnost zjednodušují, a tím usnadňují vykreslování.

Optické modely předpokládají spojitý průběh veličiny, ale data uložená na paměťovém zařízení jsou diskretizována. Kapitola 3 se věnuje rekonstrukci diskretizovaných dat na spojitou veličinu, která je uvažována v optických modelech. Druhá část této kapitoly se věnuje diskretizaci rovnic matematických modelů, aby je bylo možné numericky vyřešit.

Optické vlastnosti materiálu nejsou typicky známy a přiřazení těchto vlastností na zdrojová data se věnuje kapitola 4.

Kapitola 5 popisuje běžně využívané metody pro zobrazení objemových dat, některé optimalizační metody a stejně tak některé přístupy pro výpočet lokálního a globálního osvětlení i metody ambient occlusion, která částečně spadá do metod globálního osvětlení.

Kapitola 6 se věnuje autorskému přístupu k aplikaci a implementaci všech zmíněných poznatků. V této kapitole je popsána implementace metody ray castingu, vybraných optimalizačních metod a lokálního osvětlovacího modelu. Jsou zde také popsány dvě metody globálního osvětlení. První z nich řeší osvětlení a ambient occlusion pomocí trasování kužele a druhá – autorská metoda řeší globální osvětlení předpočítáním stínů.

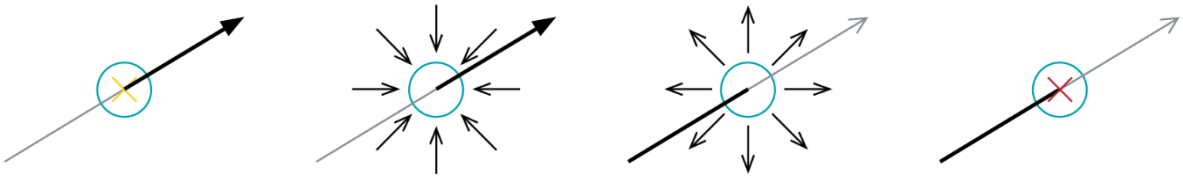
Dosažené výsledky získané z obou metod globálního osvětlení, z aplikace lokálního osvětlení a optimalizačních metod jsou prezentovány v kapitole 7.

Celkové shrnutí a další možnosti rozšíření v této oblasti jsou obsaženy v závěru.

2 Průchod světla materiálem

Volumetrická data, reprezentovaná převážně pravidelnou trojrozměrnou krychlovou mřížkou, vznikají diskretizací vypočtených nebo snímaných objektů. Snímání je realizováno medicínskými nebo jinými skenery, které přímo poskytují trojrozměrná diskrétní data. Při výpočtu je generování dat řešeno rasterizací polygonálních modelů těles. Korektní vizualizace volumetrických dat leží na fyzikálním základu průchodu světla materiálem. Světlo během průchodu materiálu s tímto materiálem interaguje. Možné jsou tři fyzikální jevy.

1. Emise – Materiál sám o sobě vyzařuje světlo. Příkladem je oheň, kde se tepelná energie přeměňuje mimo jiné na světlo, a to se vyzařuje do okolí.
2. Absorpce – Materiál absorbuje energii, která do něj vniká, a tím se utlumí energie, která z materiálu odchází. Energie paprsku je částečně nebo zcela utlumena.
3. Rozptyl – Paprsek světla interakcí s materiálem změní svůj směr. Světlo je rozptýleno do jiného nebo jiných směrů. Pokud je středem zájmu také směr paprsku, lze rozptyl rozdělit na dvě kategorie vzhledem ke směru odraženého světla.
    a. In-scattering – Přírůstky světla, které do daného bodu dopadly ze všech možných směrů a které zároveň byly odraženy do požadovaného směru.
    b. Out-scattering – Paprsek (paprsky) světla v daném bodě přicházející z daného směru, jsou odraženy do všech možných směrů.



Obrázek 1 – Zobrazení možných interakcí se světlem. Zleva: emise světla, in-scattering, out-scaterring, absorpce světla. Zdroj: [22]

Tyto tři interakce ovlivňují viditelné záření, a tím výslednou intenzitu světla dopadající k pozorovateli. Pokud jsou uvažovány všechny přírůstky a úbytky světla díky zmíněným interakcím s materiálem, lze v libovolném bodě získat intenzitu světla pro paprsek, směřující požadovaným vektorem. Výpočet celkové intenzity světla v daném bodě a v daném směru, lze rozdělit výpočet na dílčí kroky – výpočet celkové absorpce a celkové emise světla.

Celková absorpce světla je daná vztahem:

$$\chi(x, \omega) = \kappa(x, \omega) + \sigma(x, \omega)$$

kde χ je celková absorpce, κ je pravá absorpce, jak je zmíněná výše. Absorpce κ bývá také označována jako extinkční koeficient τ. Příkladem jsou černé materiály, které pohlcují sluneční světlo, a díky tomu se zvyšuje jejich teplota. Člen σ označuje out-scattering. Světlo odražené do jiného směru, než ve kterém je pozorován úbytek. Všechny tři členy rovnice jsou závislé na pozici x daného bodu a směru sledovaného paprsku ω.

Celková emise světla je dána vztahem:

$$\eta(x,\omega) = q(x,\omega) + j(x,\omega)$$

kde η je celková emise světla, to znamená intenzita světla opouštějící daný bod v daném směru. q je emise světla, jak je popsaná výše. To znamená, že materiál sám o sobě přeměňuje energii na záření, například teplo na světlo. In-scattering je dán členem j. Opět jsou všechny členy závislé na pozici x a požadovaném směru paprsku světla ω.

Členy rovnic κ, σ a q jsou optické vlastnosti materiálu. Ty mohou být dané. V případě zobrazování volumetrických data, pokud tyto vlastnosti nejsou známé, mohou být přiřazeny materiálu pomocí přechodové funkce (transfer function) viz. kapitola 4. Opakem je koeficient j. Ten není dán a je nutné ho spočítat. Pro jeho výpočet je potřeba vzít v úvahu všechny příchozí směry paprsků, které budou přispívat svojí intenzitou v daném místě. Když paprsky, ze všech směrů dorazí do daného bodu, je nutné znát pravděpodobnost jejich odrazu z příchozího směru do požadovaného směru. To vede k výpočtu:

$$j(x,\omega) = \frac{1}{4\pi}\int_{\Omega} p(x,\omega',\omega)\cdot I(x,\omega')d\omega'$$

Integrál Ω značí integraci přes kulovou plochu a je v rovnici z nutnosti započíst všechny možné směry příchozích paprsků. $p(x,\omega',\omega)$ je pravděpodobnost odrazu světla v bodě x přicházejícího ze směru ω' a odraženého do směru ω. $I(x,\omega')$ je intenzita příchozího světla v bodě x ze směru ω'. [13,23]

2.1 Optické modely

V reálném světě dochází k mnohačetným odrazům světla v závislosti na prostředí, kterým světlo prochází. Při průchodu světla materiálem dochází ke všem zmíněným interakcím. Současně se paprsek může lámat a tím vnikají další paprsky, pro které opět mohou nastávat všechny interakce. Přesný výpočet osvětlení je náročný a výpočetně složitý proces. Optické modely určují, které z interakcí mohou a které nemohou nastat. Tím se celý proces zjednoduší na úkor přesnosti. Pokud se při uvažování optických modelů uvažuje, že paprsek v průběhu průchodu tělesem nemění svůj směr, lze zaměnit parametry x a ω za parametr s, určující vzdálenost, kterou paprsek světla urazil. [13,23]

2.1.1 Čistě absorpční

Čistě absorpční model předpokládá, že těleso obsahuje pouze černé částice, které absorbují veškeré světlo, se kterým interagují. Žádné jiné interakce nejsou pro tento model přípustné. Předpokládá se, že pro malou hloubku materiálu, ve kterém jsou jednotlivé částice, je pravděpodobnost jejich vzájemného překrývání malá. Za podmínky, že hloubka materiálu

Δs se limitně blíží nule a současně pravděpodobnost překrývání jednotlivých částic v daném směru pozorování limitně blíží nule, potom intenzita světla pronikající skrz lze vyjádřit vztahem

$$I = -\kappa(s) \cdot I(s)$$

kde I je výsledná intenzita světla, $\kappa(s)$ je absorpce světla v dané délce paprsku a $I(s)$ je vstupní intenzita světla. Pro útlum světla procházející skrz celé těleso je rovnice upravena na diferenciální rovnici

$$\frac{dI}{ds} = -\kappa(s) \cdot I(s)$$

jejímž řešení je vztah

$$I(s) = I_0 e^{-\int_0^s \kappa(t)dt}$$

kde I_0 je intenzita světla vstupující do materiálu.

Tělesa zobrazené pomocí toho modelu jsou čistě černá. Zdůrazněny jsou obrysy tělesa, ale vnímání hloubky modelu je ztraceno, stejně jako jakákoliv struktura tělesa, mimo té ležící na okraji. Porovnání výsledků jednotlivých optických modelů je zobrazeno na konci kapitoly. [23]

2.1.2 Čistě emisní

Částice v tělese emitující světlo reálně světlo také pohlcují. Předpokladem je, že velikost jednotlivých částic nebo jejich počet se limitně blíží nule a současně intenzita emitovaného světlo jednou částicí se limitně blíží nekonečnu. S tímto předpokladem se absorpce světla neuvažuje a zůstává pouze emisní složka. Intenzita světla opouštějící těleso je dána diferenciální rovnicí:

$$\frac{dI}{ds} = q(s)$$

jejím řešením je:

$$I(s) = I_0 + \int_0^s q(t)dt$$

Problémem zobrazení čistě emisního modelu pro světlo dopadající směrem k pozorovateli je intenzita, jejíž hodnota může překročit hodnotu zobrazitelnou na počítači (standardně větší než 1). Využitelnost tohoto přístupu je například pro generování obrázků jako rentgenových snímků. [23]

2.1.3 Absorpčně emisní model

Kombinací dvou předchozích modelů vzniká absorpčně emisní model. Jednotlivé částice v tělesu mohou současně světlo vyzařovat, stejně tak i pohlcovat. Tím se model více blíží

reálnému světu. Rozptyl světla a nepřímé osvětlení jsou i v tomto modelu vynechány. Výpočet intenzity světla dopadající na dané pozici paprsku světla je dán vztahem:

$$\frac{dI}{ds} = q(s) - \kappa(s) \cdot I(s)$$

Úpravou a řešením rovnice pro výpočet intenzity světla v úseku paprsku uvnitř objektu vede k výrazu:

$$I(D) = I_0 e^{-\int_{s_0}^{D} \kappa(t)dt} + \int_{s_0}^{D} q(s) e^{-\int_{s}^{D} \kappa(t)dt} ds$$

Tento vztah je také označován jako integrál pro objemový rendering (z ang. volume-rendering integrál), který hraje významnou roli při vykreslování volumetrických dat. Při vykreslování tento model poskytuje kompromis mezi rychlostí vykreslování a kvalitou výsledného obrazu. Pro větší realističnost může být tento model rozšířen o jednoduchý rozptyl světla. Pro každý bod v materiálu se předpokládá, že světlo do tohoto bodu proniká skrz materiál bez interakce s ostatními body materiálu.

Rozšíření modelu dodá další prvek realističnosti, ale výsledek zcela neodpovídá reálnému zobrazení. Pro přesnější zobrazení je potřeba využít jiný model, který dále zvýší realističnost. [13,23]

2.1.4 Jednoduchý rozptyl a stíny

Přechozí modely předpokládaly pouze paprsky světla z bodů uvnitř materiálu směrem k požadovanému bodu. Rovnice tak mohly být parametrizovány pomocí parametru s, který vyjadřoval vzdálenost směrem po paprsku světla. Tento model už předpokládá více paprsků světla vlivem odrazu paprsku do více směrů. Z toho důvodu tato parametrizace už není v podobě, v jaké byla použita v předchozích modelech, možná.

Jako nejjednodušší případ tohoto modelu je stejný případ, jako pro rozšíření absorpčně emisního modelu. Pro každý bod tělesa se předpokládá, že světlo k tomuto bodu dojde od zdroje světla bez předchozí interakce s jakýmkoliv jiným tělesem nebo bodem v materiálu. Tento případ bývá také označován jako „Utah approximation“ [23].

Výpočet světla opouštějící daný bod x v daném směru ω přicházející ze směru ω' je dán vztahem:

$$I(x, \omega) = p(x, \omega', \omega) \cdot i(x, \omega')$$

$I(x, \omega)$ je intenzita světla opouštějící bod x ve směru ω. $p(x, \omega', \omega)$ je pravděpodobnost odrazu světla přicházejícího ze směru ω' v bodě x do směru ω. $i(x, \omega')$ je intenzita příchozího světla v bodě x ze směru ω'. Při zjednodušení pravděpodobnosti odrazu konstantně pro celé

těleso a pokud bude závislá pouze na úhlu mezi vektory ω a ω', lze pravděpodobnost odrazu zjednodušit na následující vztah:

$$p(x, \omega', \omega) = r(\omega \cdot \omega') \cdot a(x)$$

$a(x)$ je albedo neboli míra odrazivosti světla v bodě x a $r(\omega \cdot \omega')$ je fázová funkce. Tato funkce se typicky určuje stejně pro celé těleso. Určuje pravděpodobnost, s jakou se světlo odrazí do daného úhlu nezávisle na pozici. Proto je její parametr skalární součin vektorů ω a ω'.

Celkový vztah pro výpočet intenzity světla dopadající do daného bodu je rozšíření volume-rendering integrálu o výpočet intenzity světla z předchozí rovnice. Intenzita je pak vypočítána jako:

$$I(D) = I_0 e^{-\int_{s_0}^{D} \kappa(t)dt} + \int_{s_0}^{D} \left(\int_{\Omega} p(x(s), \omega', \omega) \cdot i(x(s), \omega') d\omega' + q(s) \right) \cdot e^{-\int_{s}^{D} \kappa(t)dt} ds$$

Vzhledem k tomu, že rovnice je parametrizována podle dráhy paprsku s, je pozice x funkcí dráhy paprsku $x(s)$. $i(x(s), \omega')$ je intenzita příchozího světla na paprsku v bodě $x(s)$ přicházejícího ze směru ω'. Výpočet intenzity spočívá ve výpočtu volume-rendering integrálu po dráze paprsku ω' směřující z/do bodu $x(s)$.

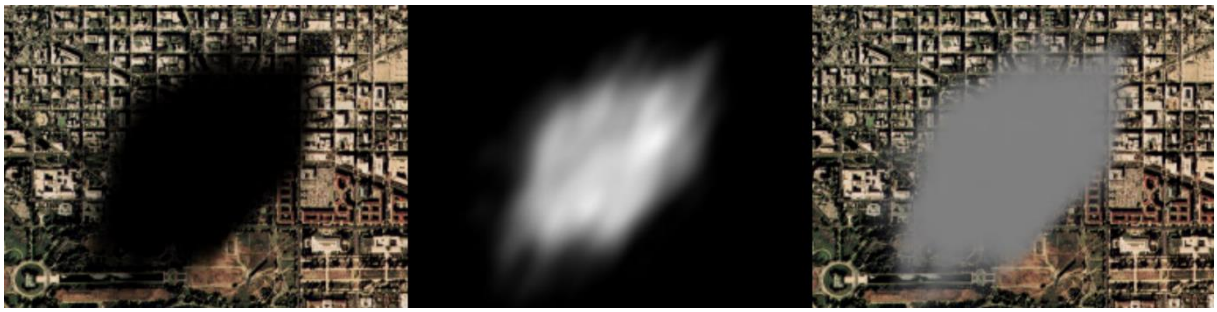
Tento model je validní, pokud se jedná o materiál, jehož albedo je nízké nebo hustota částic v něm je nízká. V těchto případech je pravděpodobnost odrazu už jednou odraženého světla malá. Jak autor uvádí, to není případ atmosférických mraků i jiných těles. Pro ty je potřeba využít ještě přesnější optický model. [13,23,24]

2.1.5 Vícenásobný rozptyl světla

Cílem modelu pro vícenásobný rozptyl světla je modelovat chování světla zahrnující všechny interakce – emisi světla, absorpci a veškerý rozptyl. Každý paprsek světla vnikající do materiálu může být pohlcen, odražen, odražen do více směrů a tím pádem se dělí na další paprsky s nižší intenzitou. Nové paprsky světla mohou být emitovány částicemi materiálu. Pro všechny paprsky, které jednou vnikly do materiálu, platí, že stejné interakce mohou nastat znovu. V teoretických podmínkách je možné, aby paprsek světla nikdy neopustil materiál a v ideálním případě může být absorpce světla nulová. V reálném světě nikoliv. Počet odrazů světla bude vždy konečný, nicméně limit na počet není dán. Implementace modelu je ze všech zmíněných modelů nejnáročnější.

Výpočet tohoto vícenásobného rozptylu světla je stejný, jako výpočet pro jednoduchý rozptyl. Rozdíl je pouze výpočet intenzity příchozího světla $i(x(s), \omega')$. Pro jednoduchý rozptyl spočívá ve výpočtu volume-rendering integrálu. Pro vícenásobný rozptyl je tento problém složitější a směřuje na rekurzivní výpočet. Pro tento model se intenzita znovu počítá stejným vztahem. [13,23,24]

Vzájemné porovnání optických modelů je zobrazeno na obrázcích 2 a 3.



Obrázek 2 – Porovnání optických modelů. Zleva: Pouze absorpční, pouze emisní, absorpčně emisní model. Zdroj: [23] – upraveno

Obrázek 3 – Porovnání optických modelů s rozptylem světla. Zleva: jednoduchý rozptyl světla, vícenásobný rozptyl světla. Zdroj [23] – upraveno

3 Rekonstrukce a diskretizace

Zmíněné optické modely, rovnice a vztahy z předchozí kapitoly předpokládají spojité veličiny. V reálném světě to platí. Aby těleso mohlo být v objemové reprezentaci uloženo, je třeba ho převést do podoby, která lze uložit na paměťovém zařízení. V praxi jsou volumetrická data získávána jako výsledek simulace nebo měření v podobě diskrétní mřížky. Typicky ve formě pravidelné mřížky, jejíž bázové vektory jsou vzájemně ortogonální. Specifickým případem pravidelné mřížky je kartézská, kde bázové vektory jsou vzájemně ortonormální – navzájem kolmé a všechny mají jednotkovou délku. Pravidelná mřížka ale není jedinou možností. Rovnoběžná mřížka je dalším typem, kde jsou bázové vektory ortogonální. Měřítko jednotlivých os však není lineární – vzdálenosti nejsou konstantní. Strukturovaná mřížka je případ, kdy jednotlivé voxely zaujímají stejný prostorový útvar (například pravidelný čtyřstěn) a jsou rozmístěny podle opakujícího se vzoru. Nestrukturovaná mřížka znamená, že voxely jsou umístěny do prostoru bez opakujícího se vzoru. Navzájem na sebe voxely navazují, ale jejich tvar ani velikost nejsou stejné. Tato práce se zabývá pouze daty uspořádanými v pravidelné kartézské mřížce.

Spojitou veličinu lze reprezentovat v pravidelné mřížce pomocí diskretizace dané veličiny, tj. vzorkování veličiny ve vybraných pozicích. Tyto hodnoty jsou snímány měřením z originálního tělesa nebo jsou získány z výsledku simulace, což je analogií pro vzorkování spojitého signálu. Celý proces se označuje jako diskretizace[1], v případě 2D a 3D dat rasterizace, resp. voxelizace. Vytváření spojitého signálu z diskrétních vzorků se označuje jako rekonstrukce.

Při zpětné rekonstrukci spojité veličiny z diskrétních dat vyvstává otázka: Lze spojitou veličinu diskretizovat tak, aby šla opět rekonstruovat do původní podoby, a pokud ano, jakým způsobem? Odpovědí na to je Nyquistův–Shannonův vzorkovací teorém. Ten říká: „Přesná rekonstrukce spojitého, frekvenčně omezeného signálu z jeho vzorků je možná tehdy, pokud byla vzorkovací frekvence vyšší než dvojnásobek nejvyšší harmonické složky vzorkovaného signálu.“ [27]. Pro data získaná nějakou simulací prováděnou třetí stranou nebo data získaná v podobě měření, zajištění vzorkování s dostatečnou frekvencí záleží na tom, kdo data vytváří, zda použije dostatečně velkou frekvenci vzorkování. [1,20]

3.1 Rekonstrukce a rekonstrukční filtry

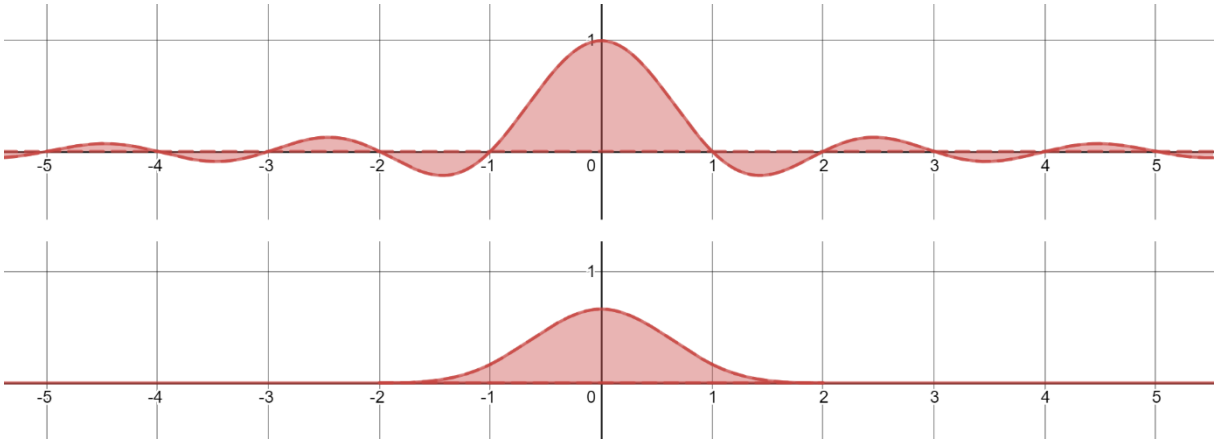
Ideální případ rekonstrukce dat ze spojité veličiny by znamenal ideálně diskretizovaná data. Data obsahující ostré hrany vyústí ve velmi vysoké frekvence spojitého signálu. Například v medicínských datech, rozdíl v intenzitě výstupního (získaného) signálu mezi kostí a tkání vedle kosti. Tyto ostré hrany v datech udávají požadavek na snímání s velmi vysokou frekvencí. Pro volumetrická data zvyšování frekvence snímání znamená rostoucí počet voxelů a jejich

[1] V rámci této kapitoly diskretizace odkazuje na dvě různé věci. První z nich je diskretizace popsaná zde. Druhá z nich je diskretizace rovnic a vztahů, kterým bude věnována pozornost dále.

zmenšující se velikost. Vzorkování ostrých hran mezi různými částmi materiálu by teoreticky znamenalo snímání s nekonečnou frekvencí. Toho není možné dosáhnout. Data s ostrými hranami pak nemohou být správně vzorkována a vzniká chyba, se kterou je nutné se smířit. Vzniká alias. Zmírnění efektu vzorkovacích chyb a alias lze do jisté míry vyřešit vhodnou rekonstrukcí signálu, a to vhodnou volbou rekonstrukčního filtru.

Rekonstrukční filtry řeší problém, jakou hodnotu je potřeba přiřadit libovolnému bodu uvnitř diskretizovaného materiálu tělesa tak, aby byla získána zpět hodnota odpovídající hodnotě v daném bodě ve snímaném materiálu se spojitým průběhem. Filtry lze rozdělit do dvou kategorií. Porovnání obou kategorií je zobrazeno na obrázku.

1. Interpolační filtry – Pro hodnoty ležící přesně uprostřed mřížky (resp. body, jejichž souřadnice jsou celočíselné), filtr vrací hodnotu odpovídající hodnotě v tomto bodě. V opačném případě se hodnota dopočítá.
2. Aproximační (vyhlazovací) filtry – Filtr přepočítává hodnoty pro všechny body nezávisle na tom, zda leží přesně v celočíselných souřadnicích mřížky nebo ne. Jinými slovy, diskretizovaná hodnota v daném bodě se po filtrování může změnit, filtr vrátí jinou než původní hodnotu.



Obrázek 4 – Porovnání interpolačního a aproximačního filtru. Nahoře: interpolační filtr sinc. Ve všech celočíselných bodech mimo bod 0 je jeho hodnota nulová. To znamená, že zachovává původní hodnotu. V ostatních bodech se hodnota dopočítá. Dole: aproximační filtr B-spline [30]. V bodě 0 není jeho hodnota 1, a tudíž nezachová původní hodnoty ani v celočíselných hodnotách mřížky, ale vyhlazuje hodnoty proložením vhodné funkce.

Rekonstrukce těchto dat je specifickým případem konvoluce. Bylo dokázáno, že sinc filtr vrací ideální jádro konvoluce. Sinc filtr je zadaný pomocí funkce sinc(x). Pro zjednodušení předpisu i pro názornost jsou zde rekonstrukční filtry definovány pro případ 1D. Nicméně lze je rozšířit na 2D i 3D. Funkce sinc je definována takto:

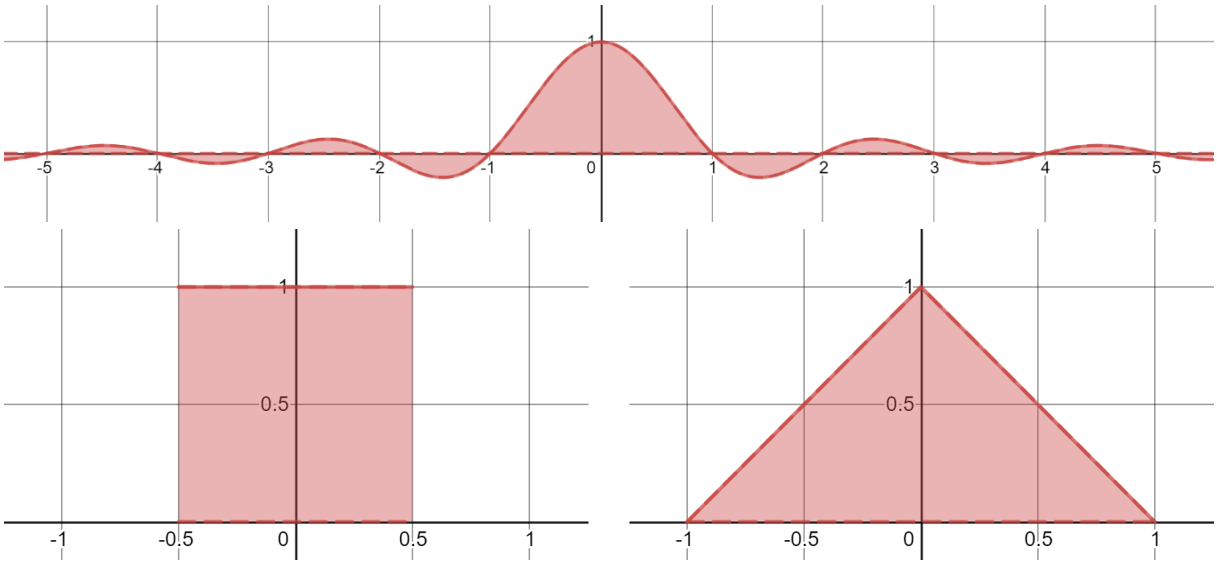
$$sinc(x) = \begin{cases} \dfrac{sin\ (x \cdot \pi)}{x \cdot \pi}, x \neq 0 \\ 1 \qquad\qquad , x = 0 \end{cases}$$

Problémem sinc filtru je jeho nekonečný rozsah. Osciluje kolem nuly přes celý svůj definiční obor. Z tohoto důvodu jsou v praxi často využívány jiné filtry. V současné době je na grafických kartách implementován box filtr a tent filtr. Tyto filtry jsou definovány následovně:

$$box(x) = \begin{cases} 1, & -0{,}5 < x \leq 0{,}5 \\ 0, & x \leq -0{,}5 \cup 0{,}5 < x \end{cases}$$

$$tent(x) = \begin{cases} 1 - |x|, & -1 < x \leq 1 \\ 0, & x \leq -1 \cup 1 < x \end{cases}$$

Grafické znázornění těchto filtrů je na obrázku 5.



Obrázek 5 – Porovnání zmíněných rekonstrukčních filtrů. Nahoře sinc filter, dole vlevo box filtr a dole vpravo tent filtr. Z grafického znázornění lze vidět, že zatímco sinc filtr má svůj roszah nekonečný, box a ten filtry jsou omezené každý na svém intervalu.

Filtry box a tent jsou filtry s dolní propustí. To znamená, že všechny frekvence překračující daný limit jsou odstraněny. Výsledný signál pak trpí ztrátou detailu. Nicméně implementace těchto dvou filtrů je přímočará a jednoduchá a vyžadují pouze nejbližší sousední vzorky. Proto jsou využívány na grafické kartě. Implementací box filtru na grafické kartě je filtrování nejbližšího souseda. To znamená, že pokud by se měla číst hodnota vzorku, který neleží na celočíselné souřadnici, jeho souřadnice se zaokrouhlí na nejbližší celočíselnou hodnotu a vrácená hodnota je jediná hodnota v jednom bodě. Filtr tent je na grafické kartě implementován pomocí lineární interpolace. V případě dvourozměrného filtru bilineární interpolace a v případě trojrozměrného filtru pomocí trilineární interpolace. Výsledná hodnota je pak součet 2 (případně 4 nebo 8) sousedních hodnot v poměru vzdálenosti k těmto vzorkům. [3, 13]

3.2 Diskretizace

Diskretizace jako pojem byl zmíněna již na začátku této kapitoly. Význam tohoto pojmu byl vzorkování spojitého signálu na diskrétní veličinu. Tato podkapitola se zabývá diskretizací rovnic optických modelů.

Značná část výpočtů během výpočtu osvětlení pro různé optické modely spočívá ve výpočtu volume-rendering integrálu. Výpočet čistě emisního nebo čistě absorpčního modelu jsou pouze specifické případy emisně-absorpčního modelu. Na druhou stranu propagace odraženého paprsku světla skrz materiál často spočívá opět v řešení tohoto výpočtu. Otázka je, proč je potřeba rovnice těchto optických modelů, nebo minimálně, proč je potřeba volume-rendering integrál diskretizovat. S použitím vhodného filtru lze určit optické vlastnosti materiálu z diskrétních hodnot jako spojitou veličinu. Nicméně analytické řešení integrálu není vždy možné najít, a pokud by bylo, řešení by často bylo velmi pracné. Namísto toho je potřeba použít vhodné numerické metody.

Běžným přístupem je rozdělení integrace na několik po sobě jdoucích intervalů. Intervaly jsou popsány pozicí na paprsku $s_0, s_1, s_2, \dots s_{n-1}, s_n$, kde $s_n = D$ (výsledná pozice) a současně platí $\forall i, j, i < j: s_i < s_j$. Původní rovnice se rozdělí na výpočet intenzit světla pronikající skrz jediný úsek paprsku. Výstupní intenzita světla v daném místě paprsku lze zapsat vztahem:

$$I(s_i) = I(s_{i-1}) \cdot T_i + c_i$$

což je zjednodušení vztahu pro volume-rendering integrál. Člen T_i představuje průhlednost materiálu ve směru paprsku na intervalu s_{i-1}, s_i a c_i je barevný příspěvek na stejném intervalu. Členy T a c nahrazují členy v rovnici volume-rendering integrálu následovně:

$$T_i = T(s_{i-1}, s_i) = e^{-\int_{s_{i-1}}^{s_i} \kappa(t)dt}$$

$$c_i = \int_{s_{i-1}}^{s_i} q(t) \cdot T(t, s_i)dt$$

Rozdělení integrálu na dílčí integrály je po menších intervalech je zobrazeno na obrázku 6. Výsledná intenzita světla lze zapsat vztahem:

$$I(D) = \sum_{i=0}^{n} \left(c_i \cdot \prod_{j=i+1}^{n} T_j \right)$$

kde $c_0 = I(s_0)$. Tento vztah lze rozepsat:

$$I(D) = I(s_n) = I(s_{n-1})T_n + c_n = (I(s_{n-2})T_{n-1} + c_{n-1})T_n + c_n = \dots$$

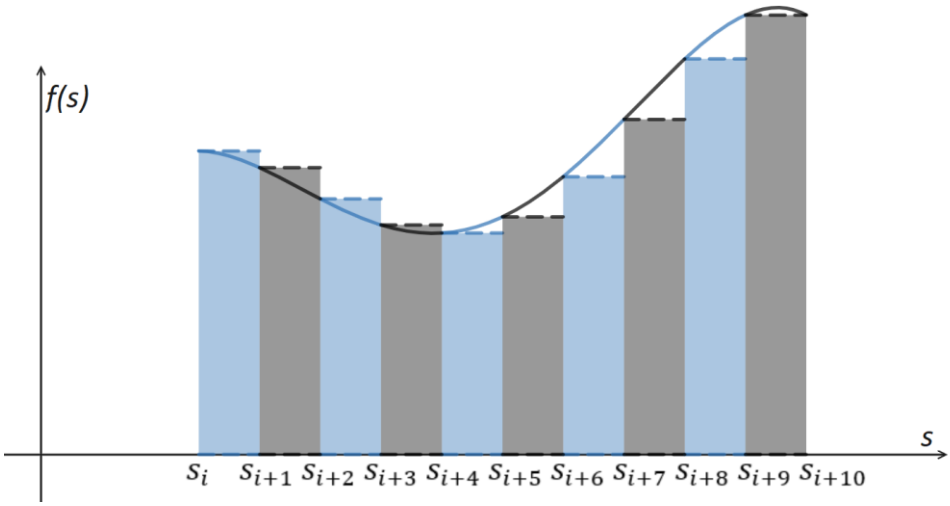
Běžný způsob aproximace řešení je pomocí Riemannovým součtem rozdělením na n ekvidistantních intervalů pomocí obdélníkové metody. Integrál v jednom intervalu odpovídá

hodnotě v jednom bodě materiálu vztažené na délku intervalu. Aproximace průhlednosti a barevné složky na daném segmentu je daná:

$$T_i \approx e^{-\kappa(s_i)\cdot\Delta x}$$

$$c_i \approx q(s_i)\cdot\Delta x$$

Δx je délka intervalu a je dána vztahem $\Delta x = (D - s_0)/n$.



Obrázek 6 – Obrázek ukazuje, že obsah plochy pod křivkou na lze spočítat jako součet obsahů ploch pod křivkou s omezením na intervalech s_{i-1}, s_i. Výpočet volume-rendering integrálu lze převést na výpočet integrálů na menších intervalech. Integrál je běžně počítán výpočtem Riemannova součtu, typicky pomocí obdélníkové metody. Integrál je pak aproximován jako součet obsahů ploch obdélníků.

Využití obdélníkové metody není jediný možná způsob. Riemannův součet umožňuje využít i jiné vhodné tvary, například lichoběžníkové pravidlo či Simpsonovo pravidlo případně i jiné pravidlo vyššího řádu podle Newton-Cotesových pravidel. Často ale volume-rendering integrál není kontinuální, a proto metody vyššího řádu často nezlepší přesnost. V praxi se namísto počítání numerického integrálu metodou vyššího řádu zvýší počet intervalů a současně s tím se sníží jejich velikost. Ne vždy je nutné, aby intervaly byly ekvidistantní. Některé části materiálu jsou téměř homogenní a počítat větší numerických integrálů nezvýší znatelně přesnost. Na druhou stranu některé oblasti tělesa jsou heterogenní a mohou obsahovat přechody mezi dvěma materiály a v tomto případě vyšší počet intervalů přesnost zvýší. Technika změny velikosti intervalů pro numerický výpočet integrálů podle homogenity nebo heterogenity v materiálu, případně podle oblasti zájmu (heterogenní oblasti jsou často oblast zájmu), se nazývá *adaptive sampling*. [13]

3.2.1 Asociované a neasociované barvy

Termíny asociované a neasociované barvy vycházejí z postupu pro skládání více obrázků přes sebe a jsou důležité z hlediska diskretizace volume-rendering integrálu. Při diskretizaci volume-rendering integrálu a při známých optických vlastnostech jsou tyto vlastnosti

reprezentovány pomocí barevného modelu, na grafické kartě pomocí RGBA. Složka A značí opacitu a přepočet mezi opacitou a průhledností je dán vztahem $\alpha = 1 - T$. Při skládání dílčích integrálů je třeba znát, jak s těmito vlastnostmi korektně pracovat.

Na počítači je pro skládání obrázků využit malířův algoritmus, kde podstatou je později přidané obrázky překreslit „nahoru“ přes aktuální obrázek, co je uložen v bufferu. Při kreslení obrázku F do bufferu B to na úrovni pixelů to znamená, že všechny pixely v B na pozicích F budou přepsány pixely obrázku F. Při dalším vykreslování obrázku G do bufferu B je postup stejný. Tento případ je ale limitován podmínkou, že všechny pixely F a G jsou zcela neprůhledné, $\alpha_F = 1$ a $\alpha_G = 1$.

Pokud chceme vyhlazovat hrany na krajích F nebo G, nebo pokud F nebo G mohou mít pixely, které jsou částečně nebo úplně průhledné, potom je třeba při jejich skládání započítat také jejich opacitu α. Při označení operace skládání barev symbolem &, operace $B\&F$ vrací novou hodnotu: $B_n = (1 - \alpha)B + \alpha F$. Problém nastává, pokud je potřeba první skládat F a G nebo pokud není zatím jasné, v jakém pořadí se budou obrázky skládat. Požadujeme tak asociativitu operace skládání:

$$(B \,\&\, F) \,\&\, G = B \,\&\, (F \,\&\, G)$$

Úpravou rovnice a aplikací vztahu pro novou hodnotu uloženou do bufferu lze postupně dojít ke vztahu

$$(F \,\&\, G) = \left(1 - \frac{\alpha_G}{\alpha_{F\&G}}\right) F + \left(\frac{\alpha_G}{\alpha_{F\&G}}\right) G$$

$$\alpha_{F\&G} = \alpha_F + \alpha_G - \alpha_F \alpha_G$$

Tento způsob lze nahradit lépe aplikovatelným způsobem. F ovlivňuje výsledek rovnice pouze, pokud je vynásobeno α_F a G je má vliv na výsledek pouze pokud je vynásobeno α_G. Myšlenka je tak předem vynásobit F a G svými opacitami. Tato reprezentace jsou *asociované barvy* – barva je vynásobena svou opacitou.

$$\widetilde{F} = \alpha_F F$$

$$\widetilde{G} = \alpha_G G$$

$$\widetilde{F\&G} = \alpha_{F\&G}(F \,\&\, G)$$

Využitím těchto definic lze získat finální tvar rovnic:

$$\widetilde{F\&G} = (1 - \alpha_G)\widetilde{F} + \widetilde{G}$$

$$\alpha_{F\&G} = (1 - \alpha_G)\alpha_F + \alpha_G$$

Pojem asociované barvy jsou barvy, jejichž barevné složky jsou předem vynásobeny vlastní opacitou. Jejich skládání se pak řídí jinými pravidly než *neasociované barvy*. Neasociované barvy jsou pak ty barvy, které svou opacitou vynásobeny nejsou. Na obrázku 7

je zobrazeno vykreslování trojúhelníku na bílé plátno, kde je pro korektní zobrazení potřeba využít asociované barvy. [6,13,28]

Obrázek 7 – Na obrázku je znázorněno vykreslování jednoho stejného trojúhelníku na bílé plátno. Vlevo: trojúhelník je vykreslen se všemi pixely plně neprůhlednými. Každý vrchol má svoji jasně danou barvu – červenou, modrou a zelenou. Uprostřed: Stejný trojúhelník je vykreslen, ale červený a zelený vrchol mají nulovou opacitu. Přechod mezi modrým a těmito vrcholy je interpolován i s barvami. Vpravo: barvy ve všech vrcholech jsou asociované. Přechod mezi modrým vrcholem ke druhým dvěma už je bez vmíšené červené a zelené složky.

3.2.2 Kompozitní schémata

Jak správně sčítat barevné příspěvky a průhlednost určují kompozitní schémata. Při práci na grafické kartě je namísto průhlednosti používaná opacita. Kompozitní schémata rozebíraná v této kapitole pro řešení volume-rendering integrálu jsou dvě: front-to-back a back-to-front. Obě schémata jsou aplikací Riemannova součtu.

Kompozitní schéma front-to-back předpokládá trasování paprsku směrem od bodu, kde je potřeba vypočítat výslednou intenzitu světla. V každém kroku trasování paprsku, to znamená pro výpočet každého integrálu na daném intervalu paprsku, jehož délka je délka kroku při trasování paprsku, je získána intenzita barevného příspěvku a nasčítaná neprůhlednost na daném intervalu. Front-to-back schéma pak tuto informaci propaguje dále po směru paprsku. Jinými slovy přepočítává, kolik světla by prošlo z následujícího intervalu směrem k prvotnímu bodu. Vztahy pro iterační výpočet barevného příspěvku jsou:

$$C_{dst} \rightarrow C_{dst} + (1 - \alpha_{dst})\alpha_i C_i$$

$$\alpha_{dst} \rightarrow \alpha_{dst} + (1 - \alpha_{dst})\alpha_i$$

Vztahy jsou zapsané ve tvaru implementovatelném na algoritmicky na grafické kartě i na procesoru. Aplikovatelné jsou na neasociované barvy. Veličiny C_{dst} a α_{dst} jsou již uložené a obsahují hodnoty nasčítané kroky z předchozích iterací. Index dst je převzatý z terminologie OpenGL, kde značí již uložené hodnoty (Obdobně terminologie OpenGL označuje indexem src nově vypočítané hodnoty, které mají být smíšeny s uloženými. Zde je tento index nahrazen pořadovým indexem podintervalu i.). C_i a α_i jsou hodnoty numerického výpočtu volume-rendering integrálu na intervalu s_{i-1}, s_i. V tomto kompozitním schématu je barevná složka označena C namísto c. Je to z důvodu, že se předpokládá implementace tohoto schématu algoritmicky a složka C je standardní třísložkový vektor barev červené, modré a zelené.

Kompozitní schéma back-to-front předpokládá trasování paprsku z nekonečna směrem do bodu, kde je potřeba výsledná intenzita zjistit. V reálném případě trasování neprobíhá z nekonečné vzdálenosti, ale trasování začíná na konci tělesa nebo materiálu případně od nějaké známe neprůhledné překážky ležící ve směru paprsku. Pro toto schéma není potřeba zjišťovat aktuální naakumulovanou opacitu nebo průhlednost. Samotná průhlednost se již promítne do výsledné barvy. Výpočet barevné složky pro toto schéma je dáno vztahem:

$$C_{dst} \rightarrow C_{dst}(1 - \alpha_i) + \alpha_i C_i$$

Tento vztah opět uvažuje neasociativní barvy.

Vztahy pro asociativní barvy následující. Pro front-to-back kompozitní schéma:

$$C_{dst} \rightarrow C_{dst} + (1 - \alpha_{dst})C_i$$

$$\alpha_{dst} \rightarrow \alpha_{dst} + (1 - \alpha_{dst})\alpha_i$$

a pro back to front kompozitní schéma:

$$C_{dst} \rightarrow C_{dst}(1 - \alpha_i) + C_i$$

Vztahy se od sebe liší pouze tím, že nové přírůstky barvy C_i se pro neasociativní barvy vynásobí opacitou dané barvy. Tímto se barva stane asociativní a nedojde k chybnému míšení barev. Viz. obrázek 7.

Existují i další kompozitní schémata, například projekce maximální intenzity (z ang. maximum intensity projection), kde výsledná barva je maximální hodnota intenzity jednotlivých integrálů. Toto schéma vytváří obrázky ve stylu rentgenových snímků, což může uživateli přinést jiný pohled na pozorovaný objekt nebo materiál. [13]

3.2.3 Opacity correction

Výpočet volume-rendering integrálu jeho rozdělením na menší intervaly je běžný způsob jeho řešení. Při uvažování pravidelné kubické mřížky, pokud délka kroku odpovídá přesně velikosti strany jednoho voxelu, řešení intergrálu pomocí Riemannova součtu je přímočará operace. Pomocí zvoleného filtru je získána hodnota v požadovaném bodě ve směru sledovaného paprsku. Tato získaná hodnota je rovna obsahu plochy pod křivkou při použití obdélníkové metody to znamená numericky nasčítané optické vlastnosti.

Častým případem je, že vzdálenost vzorků mezi sebou není rovna velikosti strany voxelu anebo se i mění v rámci jednoho sledovaného paprsku. Důvodem pro jiné vzorkování, než je podle velikosti strany voxelu je Nyquistův–Shannonův vzorkovací teorém. Ten říká, že vzorkováním spojité, frekvenčně omezené funkce, lze signál diskretizovat tak, aby byl rekonstruovatelný do původního stavu, pokud vzorkovací frekvence bude vyšší než dvojnásobek nejvyšší obsažené harmonické frekvence. Nicméně získaná data jsou již diskretizována. Výpočty nad daty ale předpokládají jejich spojitost. Zda data byla správně diskretizována nebo ne je problém, který v této fázi nelze ovlivnit. Pokud data byla vzorkována

správně (nebo minimálně dostatečně přesně), potom nejvyšší frekvence dána nejmenší vzdáleností dvou sousedních voxelů. Z toho důvodu je pak rozdělení volume-rendering integrálu na menší intervaly zvoleno tak, že jeden interval je menší než polovina velikosti voxelu. Dalším důvodem je již jednou zmíněný adaptive sampling, který může být využit kvůli optimalizaci vykreslování.

Obdélníková metoda Riemannova součtu, pokud délka intervalu je rovna celé délce integrálu, počítá s proměnnou optickou vlastností materiálu v integrálu tak, jako kdyby byla konstantní funkce. Při uvažování této metody lze pak průhlednost vypočítat vztahem:

$$T = e^{-\int_{s_{i-1}}^{s_i} \kappa(t)\,dt} = e^{-\int_{s_{i-1}}^{s_{i-1}+\Delta x} \kappa(t)\,dt} \approx e^{-\int_{s_{i-1}}^{s_{i-1}+\Delta x} \kappa\,dt} = e^{-\kappa\cdot\Delta x}$$

Δx je délka intervalu, která odpovídá vzorkovací frekvenci použité při vytváření diskrétních volumetrických dat. To znamená, že je rovna vzdálenosti dvou sousedních voxelů. $\widetilde{\Delta x}$ je upravená délka intervalu. Tato délka odpovídá vzorkovací frekvenci při rekonstrukci signálu. Upravená průhlednost se vypočte vztahem:

$$\tilde{T} = T^{\frac{\widetilde{\Delta x}}{\Delta x}} \Rightarrow \tilde{\alpha} = 1 - (1-\alpha)^{\frac{\widetilde{\Delta x}}{\Delta x}}$$

Mocnina $\frac{\widetilde{\Delta x}}{\Delta x}$ lze jinými slovy vyjádřit jako velikost úseku paprsku lomeno délka strany jednoho voxelu při konstantní velikosti voxelů. Barevná složka počítaná vztahem $c_i \approx \mathrm{q}(\mathrm{s_i}) \cdot \Delta\mathrm{x}$ se změnou vzorkovací frekvence změní také. Nově vypočítaná barevná složka je pak dána vztahem

$$\tilde{c} = c \cdot \frac{\widetilde{\Delta x}}{\Delta x}$$

[13,18]

4 Přechodová funkce

Výpočty a rovnice popisované v předchozí kapitole vyžadují znalost vlastností materiálu, jako je emisní koeficient a absorpční koeficient pro každý bod objemu. Pokud dané koeficienty nejsou zadané dopředu, je potřeba zajistit jejich přiřazení pro každý bod materiálu podle jiných vlastností.

Vědecká volumetrická data jsou většinou zadaná ve formě skalárních hodnot pro každý vzorek materiálu. Tyto naměřené vzorky přímo neurčují požadované koeficienty nutné pro výpočet. Přiřazení těchto hodnot pak není jednoznačné a určí se podle uživatelem zadané funkce. Právě funkce, která přiřadí optické vlastnosti pro body materiálu podle zadaných skalárních hodnot, se označuje přechodová funkce (z ang. transfer function). Pro zobrazení těchto hodnot na počítači tato funkce mapuje vstupní intenzity materiálu na barvu a opacitu. Při implementaci jsou optické vlastnosti nahrazeny barevnou složkou a opacitou.

Přechodové funkce lze rozdělit podle několika kritérií. Podle počtu vstupních parametrů na jeden bod materiálu lze přechodové funkce rozdělit na jednorozměrné, dvourozměrné, třírozměrné atd. Přechodové funkce dvourozměrné a více lze označit jako multidimenzionální. Čím více skalárních hodnot vycházejících z různých měření na jeden bod v materiálu je k dispozici, tím přesnější je přiřazení optických vlastností pro body materiálu. S narůstajícím počtem rozměrů přechodové funkce se stává přiřazování optických vlastností náročnější záležitostí. Cílem je nalezení přechodové funkce, která bude mapovat optické vlastnosti materiálu tak, aby vynikly důležité oblasti materiálu, zatímco na aktuálně nedůležité oblasti nebyla poutána pozornost uživatele, nebo tak, aby výsledný obraz co nejvíce odpovídal reálnému tělesu. To je obtížné, a proto je to ponecháno na volbě uživatele, který musí sám určit, co je pro něj oblast zájmu a co nikoliv.

Data v diskrétní mřížce mají skalární hodnoty materiálu uloženy pouze na pozicích odpovídajících celočíselným souřadnicím. Optické vlastnosti materiálu mimo celočíselné souřadnice se musí podle využitého filtru dopočítat. Podle toho, v jakém kroku se přechodová funkce na data aplikuje, se rozlišují typy přechodových funkcí z hlediska klasifikace. [5,13]

4.1 Pre, post a před-integrovaná klasifikace

Prvním typem je před-interpolační neboli před-klasifikační přechodová funkce. Před-interpolační přechodová funkce znamená, že se funkce na data aplikuje dříve než rekonstrukční filtr. Jinými slovy by se dalo chápat, že pro veškeré hodnoty materiálu na diskrétních pozicích jsou předpočítány a rekonstrukční filtr je použit až na přiřazené optické vlastnosti.

Druhým typem je post-interpolační nebo post-klasifikační přechodová funkce. Post-interpolační přechodová funkce znamená, že se tato funkce na data aplikuje až poté, co se použije rekonstrukční filtr.

Oba tyto přístupy s sebou mohou přinést nové vysoké frekvence signálu. To je případ, pokud má přechodová funkce velký sklon – první derivace pro přechodovou funkci má v daném místě velkou funkční hodnotu. Tyto případy nejsou ojedinělé. V medicínských datech přechod mezi tvrdou tkání a měkkou tkání je jeden z případů, kde se typicky využívá přechodové funkce s velkým sklonem. Zvýšený počet vzorků řeší tento problém ale na úkor výkonu vykreslování. Často tak jsou pozorovány artefakty vzniklé nedostatečným vzorkováním, pokud nemá přechodová funkce velmi mírný průběh. Pro překonání této překážky, je potřeba lépe aproximovat volume-rendering integrál. Způsobů bylo navrženo několik, například využití numerických integračních metod vyššího řád (lichoběžníkové pravidlo Riemannova součtu, Simpsonovo pravidlo, ...), využití adaptive samplingu, nicméně tyhle metody neřeší samotný problém vysokých frekvencí přechodových funkcí. Metoda před-integrované klasifikace (nebo také před-integrovaná přechodová funkce) tento problém řeší tím, že vyhodnocuje volume-rendering integrál pomocí přechodové funkce ne v rámci jednoho bodu, ale v rámci dvou bodů na přímce. Metoda před-integrované přechodové funkce navrací optické vlastnosti materiálu v bodě na paprsku v závislosti na třech parametrech. Skalární hodnota na začátku úseku paprsku s_f, skalární hodnota na konci úseku paprsku s_b a délka úseku d. Pokud je délka všech úseků paprsku stejná, d je konstantní a navrácená hodnota je závislá pouze na s_f a s_b, opacita a barevná složka jsou vypočítány následujícím způsobem:

$$\alpha_i \approx 1 - e^{-\int_0^1 \kappa((1-\omega)\cdot s_f + \omega \cdot s_b)\cdot d \cdot d\omega}$$

$$C_i \approx \int_0^1 q\left((1-\omega)\cdot s_f + \omega \cdot s_b\right) \cdot e^{-\int_0^\omega \kappa((1-\omega')\cdot s_f + \omega' \cdot s_b)\cdot d \cdot d\omega'} \cdot d \cdot d\omega$$

Před-integrovaná přechodová funkce vrací vždy asociované barvy, což hraje roli při použití ve vykreslovacím procesu při skládání jejích hodnot. Nevýhodou tohoto přístupu je nutnost integrovat přes velký počet segmentů pro každou novou přechodovou funkci. To by zabraňovalo interaktivní změně přechodové funkce. Byl proto navržen jiný přístup. Výpočet α_i a C_i lze aproximovat následujícím způsobem:

$$\alpha_i \approx 1 - e^{-\frac{d}{s_b - s_f}\left(T(s_b) - T(s_f)\right)}$$

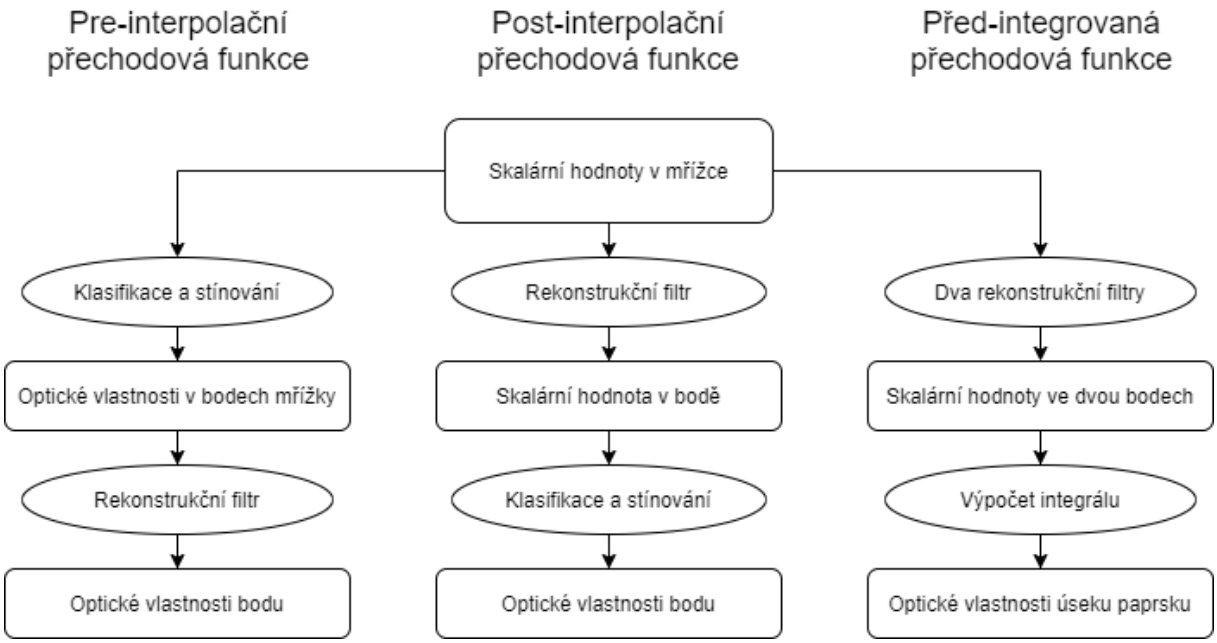
$$C_i \approx \frac{d}{s_b - s_f}\left(K(s_b) - K(s_f)\right)$$

kde funkce T a K jsou definovány:

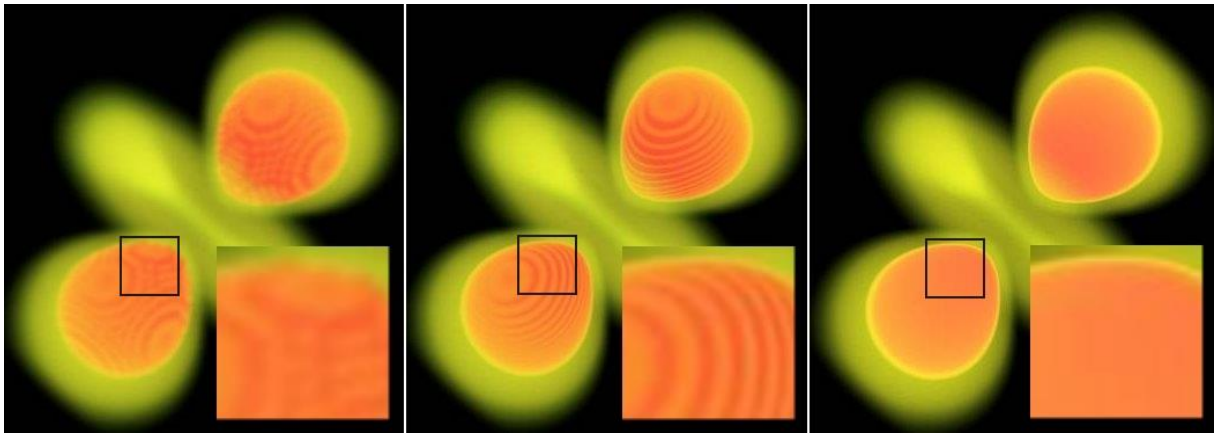
$$T(s) = \int_0^s \kappa(s')ds'$$

$$K(s) = \int_0^s q(s')ds'$$

Základní rozdíl mezi těmito třemi přístupy je uveden na obrázku 8. Rozdíl ve výsledcích jednotlivých metod je pak zobrazen na obrázku 9. [5,13,16,21]



Obrázek 8 – Porovnání rozdílu mezi třemi zmíněnými přístupy na aplikaci přechodové funkce.



Obrázek 9 – Porovnání výsledků třech zmíněných přístupů. Vlevo: pre-klasifikační přechodová funkce. Artefakty jsou velmi znatelně viditelné. Snadno lze vidět strukturu mřížky. Uprostřed: post-klasifikační přechodová funkce. Stále jsou zřetelné artefakty vzniklé nedostatečným vzorkováním. Vpravo: před-integrovaná přechodová funkce. Díky výpočtu přechodové funkce pro úsek paprsku, artefakty v obrázku nejsou. Zdroj: [5]

5 Zobrazovací metody

Zobrazení volumetrických dat bylo vždy výpočetně náročným procesem. Oproti povrchovým modelů, které lze podle použitého algoritmu vykreslit ihned přímo, u volumetrických dat je oproti tomu potřeba vykreslit nikoliv souvislý povrch ale souvislý objem. V průběhu zkoumání nových metod a také díky vzrůstu výpočetního výkonu grafických karet, ale i procesorů a také kapacity pamětí a přenosových rychlostí sběrnic, byly vytvářeny nové metody zobrazení. Vzrůst výkonu počítačů umožnil přímé vykreslování volumetrických dat (direct volume rendering).

Jedna z metod využívaných k přímému vykreslování objemů je plátkování. Tato metoda využívá funkcionalitu grafické karty stylem, pro který je grafická karta určena, a to k vykreslování povrchu modelů. Je kombinací vykreslování povrchu a těles a objemu v tělese. Tato metoda má několik modifikací a dále může být rozšiřována. Základní princip je ale stejný. Každé objemové těleso má svoji obálku (bounding box), uprostřed které se nachází. Objemový model, který se má vykreslit je uložen v grafické kartě ve formě textury. Při vykreslování se skrz tuto obálku provedou řezy. Na každý řez se mapuje uložená textura. Vybere se vhodné kompoziční schéma (front-to-back, back-to-front) a pro každý rozrastrovaný pixel se toto schéma použije a výsledek se uloží zpět výsledného obrazu. Existují varianty tohoto přístupu. Lze je rozdělit podle způsobu uložení i způsobu vykreslování. Jedním způsobem je varianta, která se využívala převážně v době, když grafické karty neumožňovaly práci s trojrozměrnými texturami. Materiál se uloží do několika po sobě jdoucích dvojrozměrných textur. V současné době grafické karty práci s trojrozměrnými texturami umožňují. Nicméně tento přístup lze stále použít, například kvůli optimalizaci práci s texturami.

Další varianty určují, jak se jednotlivé řezy budou orientovat. Bounding-box oriented přístup říká, že se plátky orientují vzhledem k jedné ze souřadnicových os své obálky. V kombinaci s plátky uloženými ve dvojrozměrných texturách je vykreslování přímočarý proces, kdy se plátky vykreslí podle pořadí. Pokud je plátků stejně jako je počet textur, potom stačí využít pouze dvojrozměrný rekonstrukční filtr. Pro další zlepšení kvality byly navrženy metody, kde se řezy orientovaly podle osy obálky, která svírala s pohledovým vektorem nejmenší úhel. Obě tyto metody lze implementovat s míšením barev, které už je na grafické kartě implementováno. Další rozvoj ke zlepšení vizuálního kvality obrazu je orientování řezů kolmo na vektor pohledu. Na rozdíl od předchozích metod, tato metoda vyžaduje trojrozměrný rekonstrukční filtr. Využití metody plátková jsou například práce [9].

I přes to, že přímé vykreslování pomocí plátkování má značná pozitiva jako je například vysoká přenosová rychlost mezi pamětí textur a rasterizační jednotkou a vysoká rychlost rasterizace, má tato metoda i nevýhody, a to hlavně pro velké data sety. Počet jednotlivých plátků je dán velikostí objemu i mírou plátkování (Nyquistův–Shannonův vzorkovací teorém) nebo i například vzdáleností od pozorovatele. Metoda plátkování rasterizuje veškerá objemová data. Práce [17] ukazuje, že pouze malá část rasterizovaných fragmentů má vizuální dopad na výsledný obraz. Počet finálně viditelných fragmentů při zobrazení zubu z datového

souboru lidské hlavy se pohyboval v rozpětí 0,2% až 4%. To znamená, že pouze jeden z 25, v horším případě jeden z 500, rasterizovaných fragmentů byl zobrazen. [13,25,29]

5.1 Raycasting

Implementační část této práce využívá jinou metodu přímého zobrazování objemových dat. Tou je raycasting. Tato metoda má uplatnění jak při vykreslování povrchových modelů, tak při vykreslování objemových dat. V obou případech se předpokládá, že od zdrojů světla k pozorovateli dorazí pouze ty paprsky světla, které lze zpětně od pozorovatele k těmto zdrojům světla trasovat. Pro povrchové modely je třeba vyřešit průnik paprsku s geometrií scény.

Pro volumetrická data se v metodě raycasting aplikuje volume-rendering integrál přímo na objemová data, v základní formě je tato metoda přímou implementací absorpčně-emisního modelu. Princip metody je následující:

1. Vypočítat směr a pozici trasovaného paprsku. Pozice pozorovatele je známá a pro každý pixel obrazu, pro který chceme spočítat volume-rendering integrál, je potřeba určit směr od pozorovatele.
2. Určit krok, po kterém se bude paprsek iterovat. V kapitole 3.2 bylo ukázáno řešení volume-rendering integrálu jeho dělením na menší podintervaly. Délka kroku odpovídá délce podintervalu integrálu.
3. Určit rekonstrukční filtr. Při implementaci na grafické kartě se nabízí možnost box a tent, ale nejsou to jediné možnosti.
4. Stanovit cyklus, který iteruje po směru paprsku od jeho začátku (průnik do materiálu / obálky tělesa) dostatečně dlouho po definované délce kroku.
    a. V každém kroku cyklu pomocí zvoleného filtru a přechodové funkce získat hodnotu v požadovaném bodě objemu tělesa. Získaná hodnota odpovídá Riemannovu integrálu a pomocí kompozičního schématu se načte k aktuálně naakumulované hodnotě.
5. Výslednou barvu pro každý paprsek zobrazit v daném pixelu, pro který byl počítán.

Stále platí, že typicky pouze malá část všech integrálů na podintervalech celého paprsku ovlivní výslednou barvu pixelu, pro který byl paprsek trasován. Výjimkou jsou materiály jako atmosférické mraky, jejichž hustota částic je malá nebo jejich částice jsou minimálně neprůhledné. V takovém případě je pak vyhodnocování integrálu zbytečné pro většinu bodů paprsku. Při využití filtrování lineární interpolací nebo filtrování nejbližšího souseda, vyhodnocení spočívá v přečtení informace z uložené textury a podle kompozitního schématu složení barevných příspěvků. Všechny tyto operace, které nemají vliv na výsledek vykreslování, celou metodu zpomalují. V dalších podkapitolách jsou popsané vybrané metody optimalizace. [12,13,15]

5.1.1 Early-ray termination

První ze zmíněných optimalizačních metod předčasně ukončuje trasování paprsku. Průchod paprsku skrz materiál může být zastaven v moment, když při trasování paprsku od pozorovatele směrem do materiálu je integrálem kumulovaná opacita 1. Žádné další barevné příspěvky nebudou mít vliv na obarvení pixelu. Typicky se diskrétní trasování ukončuje dřív, pokud se kumulovaná opacita dostatečně blíží 1. Opacita 1 by znamenala absolutně neprůhlednou část materiálu.

Trasování paprsku lze také ukončit, pokud opustí svoji obálku (bounding box). To i v případě, pokud je součástí scény, ve které jsou další polygonální modely. Řešení dalšího míchání barvy naakumulované paprskem a barevných příspěvků polygonálních modelů je typicky řešeno zobrazovacím algoritmem, a nikoliv dalším trasováním paprsku procházejícím objemem. [13,17]

5.1.2 Adaptive sampling

Metoda adaptive sampling byla zmíněna v kapitole 3.2. Materiál typicky obsahuje oblasti s vysokou mírou homogenity a současně také obsahuje oblasti, které jsou heterogenní. Velké intervaly pro výpočet integrálu v heterogenních oblastech snižují přesnost jeho výpočtu, pro homogenní oblasti to neplatí. U homogenních oblastí výpočet velkého počtu integrálů zvyšuje časovou náročnost vykreslování bez zvýšení přesnosti.

Míra homogenity a heterogenity materiálu v jeho různých částech a často je závislá na výpočtu předem vypočteného objemu – *importance volume*, ve kterém je pro každý bod materiálu uložena minimální vzdálenost vzorku podle homogenity materiálu. V průběhu vykreslování, pro každý bod trasování paprsku je současně sebrán vzorek z importance volume, kterým se určí další délka intervalu původního paprsku. Tento nově vypočítaný objem nemusí mít stejné rozměry jako původní objem. Snížení velikosti každé strany objemu na $1/2$ sníží paměťovou náročnost na $1/8$. V případě velkých datasetů není výjimkou, že velikost celého dalšího objem přesáhne kapacitu paměti. [13]

5.1.3 Empty space leaping

Metoda empty space leaping neboli metoda přeskakování volného prostoru je částečně podobná metodě adaptive samplingu. Používá se k přeskakování prostoru, který je plně průhledný. Ve volumetrických datech je častý případ toho, že zkoumaná / důležitá část dat je ve středu mřížky, zatímco okolo jsou zcela průhledné voxely. Zdrojem těchto prázdných míst uvnitř objemu jsou například metody snímání, kdy snímaný objekt leží v prostoru, kde je snímán, ale snímaný prostor je větší než těleso. Pokud dojde k ořezání objemu tělesa tak, aby na každé souřadnicové ose byly prázdné oblasti odstraněny, stále těleso nevyplní veškerý objem. Pro příklad, pokud by byl data set lidské hlavy vytvořený medicínským skenem aproximován jako koule, potom počet voxelů mimo hlavu by byl 47,64%.

Otázka tedy zní, proč by bylo potřeba iterovat v cyklu raycastingu přes oblasti s prázdnými voxely. Na to, aby bylo možné tyto prázdné oblasti přeskočit, je potřeba znát přechodovou funkci. Díky ní pak lze určit, které oblasti jsou neprůhledné a lze je přeskočit. Určení oblastí, které jsou průhledné, lze vytvořením dalšího objemu, ve kterém jsou uloženy vzdálenosti k nejbližšímu alespoň částečně neprůhlednému voxelu. Znovu jako u adaptive samplingu platí, že vytvoření objemu, jehož velikost je stejná, jako velikost původního objemu, není vždy žádoucí. Je potřeba uvažovat, co je priorita při optimalizaci. Přesnost objemu, která může zvýšit výkon vykreslování, protože přesnější délky kroků povedou k menšímu počtu kroků, nebo paměťové nároky, které rostou se třetí mocninou velikosti strany.

Řešení, které dokáže omezit paměťové nároky o značnou část je implementace přístupu dělení objemu metodou octree. Paměťová náročnost v práci [17] sekundárního objemu klesla na $1/512$ využité paměti primárním objemem. Primární objem je rozdělen na bloky velikosti 8x8x8 a pro každý blok jsou uloženy pouze dvě hodnoty, značící minimální a maximální skalární hodnoty primárního objemu. Podle těchto hodnot a známé přechodové funkce lze určit, zda lze celý tento blok přeskočit ve vykreslování kompletně, nebo je nutné přes něj iterovat. Tento přístup neumožňuje přeskočit celou prázdnou oblast až do maximální „bezpečné“ vzdálenost, během které nehrozí přeskočení alespoň částečně neprůhledného místa. Nicméně není potřeba celý sekundární objem znovu přepočítávat při změně přechodově funkce a paměťové nároky jsou zanedbatelné oproti primárnímu objemu. [13,17]

5.2 Artefakty vykreslování

Od začátku snímání dat v reálném světe do digitální podoby, přes jejich rekonstrukci filtrováním, dále při klasifikaci nebo finální numerické integraci, ve všech těchto dílčích krocích mohou vznikat (a také vznikají) nežádoucí artefakty. Při snímání jsou artefakty způsobeny malou frekvencí vzorkování. Tu ale nelze jednoduše zvyšovat do nekonečna a ostré přechody materiálu jsou typická místa, kde by vzorkovací frekvence musela jít do velmi vysokých hodnot. Při rekonstrukci vznikají artefakty použitím filtru, který nedokáže rekonstruovat správně spojitou veličinu. I když bylo dokázáno, že sinc filtr je ideální rekonstrukční filtr, jeho šířka je nekonečná a v praxi nelze bez vhodné úpravy použít. Při numerické integraci vznikají artefakty nedostatečným vzorkováním a omezenou přesností barevných kanálů.

Využití nejpřesnějších metod generujících vysokou výslednou kvalitu obrazu je sice žádoucí ale současně není vždy možné. Přesnější metody většinou přinášejí náročnější výpočty. Pro vykreslování v reálném čase je snaha o vykreslování s přijatelným počtem snímků za sekundu a současně s dostatečnou kvalitou výsledného obrazu. Je to kompromis mezi rychlostí a kvalitou. [13]

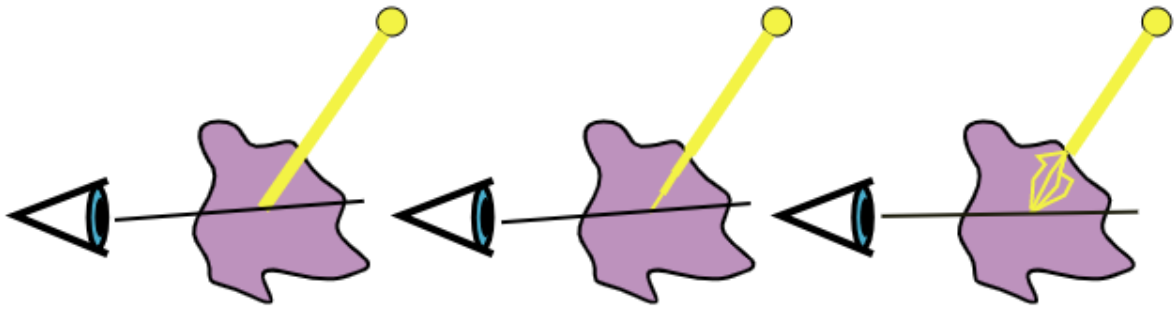
5.3 Lokální a globální osvětlovací model

Vykreslování volumetrických dat pomocí emisně-absorpčního modelu bez dalšího rozšíření nepodává výsledky, které by v současné době dostačovaly standardu. Tento model

nedokáže model vykreslit tak, aby byly jasně znatelné tvary a hrany. Rozšíření, které toto dokážou, jsou použití lokálního a globálního osvětlovacího modelu.

Využití osvětlovacích modelů je důležitým krokem k porozumění obsahu vykreslovaného tělesa a hloubkové informace. Studie prováděná v roce 2017 porovnávala vliv osvětlovacích modelů na schopnost uživatelů porozumět (umět si představit) zobrazovaná tělesa a na schopnost odhadnout pomocí obrazu hloubkovou informaci. Ve studii byly testovány tři různé přístupy. První z přístupů byl pozorovaní vykreslovaných těles bez jakéhokoliv osvětlení. Druhým přístupem bylo použití Phongova osvětlovacího modelu (viz. kapitola 5.3.1.3) spadajícího do lokálního osvětlovacího modelu. Třetím přístupem bylo využití pokročilého globálního osvětlovacího modelu. Není překvapivý výsledek, že využití osvětlovacího modelu zlepšuje schopnost vnímat hloubkovou informaci tělesa. Nejlepší výsledky podává využití globálního osvětlovacího modelu (ve studii využití měkkých stínů). [10]

Rozdíl mezi těmito modely je v jejich přístupu k dopadajícím paprskům světla do materiálu a každý z těchto modelů je vysvětlen ve vlastní podkapitole. Rozdíl mezi těmito modely je zobrazen na obrázku 10. Každý z těchto osvětlovacích modelů lze zařadit do jedné nebo více kategorií optických modelů. [13,24]



Obrázek 10 – Ukázka principu lokálního a globálního osvětlovacího modelu. Vlevo: Lokální osvětlovací model. Světlo se od zdroje světla dostane až k aktuálně osvětlovanému bodu bez interakce s materiálem. Uprostřed: Globální osvětlení – světlo interaguje s materiálem, než se dostane k osvětlovanému bodu, díky čemu se může jeho intenzita například oslabit. Vpravo: Další ukázka globálního osvětlení. V tomto případě se jedná o přesnější model, který počítá i s rozptylem světla uprostřed materiálu, než se světlo dostane k pozorovateli.

5.3.1 Lokální osvětlovací model

Lokální osvětlovací model je jedno z rozšíření emisně-absorpčního modelu o jednoduché osvětlení. Jak již bylo zmíněno, emisně-absorpční model neuvažuje rozptyl světla. Ten však dodá vykreslovanému materiálu realističnost.

Základní vlastnost lokálního osvětlovacího modelu je předpoklad, že do každého bodu materiálu dopadne paprsek světla od jeho zdroje bez interakce se zbytkem materiálu. Jakkoliv by se dalo říct, že předpoklad lokálního osvětlovacího modelu je naivní, je přesto používán. Oproti globálnímu osvětlovacímu modelu, který je popsán dále, je jeho implementace značně jednoduší právě díky tomuto naivnímu předpokladu. Vlastnosti povrchu, jako jeho hrubost a

nerovnosti by byly velmi těžko rozeznatelné bez jakéhokoliv osvětlení. Tento model právě tyto nerovnosti dokáže dobře zviditelnit. Často mnohem lépe než složitější globální model.

Model rozšiřuje volume-rendering integrál následovně:

$$I(D) = I_0 e^{-\int_{s_0}^{D} \kappa(t)dt} + \int_{s_0}^{D} \left(\left(q(s) + \sum_{i=1}^{k} (p(s, \omega_i') \cdot I_i) \right) e^{-\int_{s}^{D} \kappa(t)dt} \right) ds$$

Člen rovnice $p(s, \omega_i')$ značí pravděpodobnost, že v místě objemu na dráze paprsku v pozici s dojde k odrazu světla po směru tohoto paprsku z příchozího směru ω_i' pro všechny zdroje světla $1,2, \dots k$. I_i je intenzita daného zdroje světla. Vztah je obdobou pro výpočet intenzity pro jednoduchý rozptyl světla. Integrál přes plochu kulové plochy je nahrazen součtem pro všechny zdroje světla.

Pokud bychom chtěli aplikovat lokální osvětlení na polygonální modely, a tudíž by se neřešil tento integrál, součin $p(s, \omega_i') \cdot I_i$ by mohl být nahrazen BRDF – bidirectional reflectance distribution function. BRDF je funkce, která určuje intenzitu světla odraženého od materiálu v daném bodě do daného směru přicházející ze zdroje světla z příchozím směru. BRDF existuje značné množství. Některé jsou založeny na reálném fyzikálním základu, některé na zjednodušeném fyzikálním modelu (obdobně jako optické modely průchodu světla materiálem zjednodušují skutečnost na pouze některé fyzikální jevy, stejně tak některé BRDF zjednodušují popisovanou skutečnost). BRDF mají společné, že jsou závislé na normálovém vektoru povrchu, pro který se počítají.

Aplikace BRDF se příhodně nabízí u výpočtu volume-rendering integrálu s lokálním osvětlením pro součin členů rovnice $p(s, \omega_i') \cdot I_i$. Numerická integrace integrálu vede k výpočtu Riemannova součtu. Pro každý nasčítávaný vzorek je potřeba aplikovat BRDF.

Volumetrická data v tomto ohledu naráží na překážku. Snímaná data jsou uložena ve formě mřížky, kde v každé buňce je uložena skalární hodnota (nebo více skalárních hodnot). V předchozích kapitolách už však bylo zmíněno, že se u nich očekává spojitý průběh napříč celými daty. Některá data mohou mít ostré hrany, ale problém zůstává určit tuto hranu, aby se na ni právě BRDF dala aplikovat. Dalším problémem je, že tyto ostré hrany data obsahovat nemusí anebo materiál může mít části s plynulým přechodem.

Tradičně jsou tyto problémy řešeny pomocí výpočtu gradientu (sklonu) plochy. Předpokládá se, že světlo je odráženo od materiálu z izoploch. Tato plocha je určena svým normálovým vektorem a díky tomu ji lze hned využít ve výpočtech pro BRDF. Problémem je, že u dat nelze vždy rozlišit, co je a není hrana materiálu a jestli je ostrá nebo ne. V homogenních částech materiálu žádné izoplochy nejsou. Na ostré hraně dvou homogenních částí materiálu by naopak izoplocha měla být znatelná. V oblastech, kde skalární hodnota v materiálu se postupně mění, by izoplocha měla být kolmá na směr této změny, nicméně měla by být rozlišena od ostré hrany. Je proto vhodné počítat směr gradientu pro všechny body a současně mít možnost určit jeho „důležitost“. Jinými slovy určit, na jak ostré hraně tato

izoplocha leží. Gradient je vektor určující normálu izoplochy a jeho délka určuje jeho „důležitost“. Určit gradient lze několika způsoby. [24]

5.3.1.1 Před-počítání gradientu

Metoda spočívá ve výpočtu gradientu pro každý bod objemu předem a uložení těchto před-počítaných gradientů společně s daty. Typicky se pracuje s celočíselnými souřadnicemi. Jedním z přístupů je výpočet centrálních diferencí. Pomocí Taylorova polynomu lze funkční závislost aproximovat pomocí polynomické závislosti pro daný bod a jeho malé. Pro zjednodušení situace je zde uvažován pouze 1. rozměr. Aplikací Taylorova polynomu lze aproximovat funkční hodnotu v sousedním bodě mřížky v okolí zadaného bodu takto:

$$f(x+h) = \sum_{n=0}^{\infty} \frac{f^n(x)}{n!} \cdot h^n$$

Využití ale spočívá ve zjištění první derivace. Funkční hodnoty nulté derivace funkce jsou známé. Vhodným ukončením Taylorova polynomu předčasně získáme na pravé straně rovnice derivace původní funkce a vhodnou úpravou rovnice lze pak získat první derivace funkčních hodnot. Ukončení rozvoje polynomu po druhé mocnině se jeví jako vhodná varianta. Pro celočíselné souřadnice mřížky platí: $h = \pm 1$. Ukončené rozvoje Taylorova polynomu lze pak zapsat takto:

$$f(x+1) = f(x) + f'(x) + f''(x)/2 + o(1^3)$$
$$f(x-1) = f(x) - f'(x) + f''(x)/2 + o(1^3)$$

$o(1^3)$ je aproximační chyba třetího řádu vzniklá ukončením rozvoje Taylorova polynomu. Úpravou těchto dvou rovnic je neznámá druhá derivace odstraněna. Centrální diference je pak získána finálním vztahem:

$$f'(x) = \frac{f(x-1) + f(x+1)}{2} + o(1)$$

Při rozšíření tohoto vztahu to trojrozměrného prostoru pro výpočet centrálních diferencí dostaneme tento vztah:

$$\nabla f(x,y,z) \approx \frac{1}{2} \begin{pmatrix} f(x+1,y,z) - f(x-1,y,z) \\ f(x,y+1,z) - f(x,y-1,z) \\ f(x,y,z+1) - f(x,y,z-1) \end{pmatrix}$$

Je patrné, že pro výpočet gradientu se využije 6 okolních vzorků okolo centrálního vzorku.

Existují další metody určující před-počítaný gradient s vyšší přesností. Jednou z nich je počítání pomocí konvoluce. Centrální diference jsou jeden z případů jádra konvoluce. Větší jádro konvolučního filtru zvýší přesnost. Běžným případem jsou jádra velikosti 3x3x3, 5x5x5 ale výjimkou nejsou ani větší. Se zvětšující se velikostí roste přesnost výpočtu na úkor jeho doby trvání. Před-počítání gradientu s sebou nese tu výhodu, že vlastní výpočet může být

časově náročný, ale při vykreslování už další výpočty prováděny nejsou. Jednou z konkrétních možností pro výpočet gradientu je aplikace konvoluce pomocí Sobelova operátoru. Tento operátor se používá pro detekci hran ve dvojrozměrných obrázcích a lze ho rozšířit do třetího rozměru a tím bude detekovat místo hran izoplochy. [13]

5.3.1.2 Počítání gradientu na vzorku

Postup počítání gradientu na vzorku se liší oproti předchozímu tím, že data nejsou předpočítána a uložena do dalšího objemu, ale jsou počítána pro každý vzorek, což je pomalejší. Hlavně v případě filtrů s větším konvolučním jádrem se rozdíly začnou projevovat nejvíce. Všechny postupy aplikovatelné pro počítání gradientu v přípravné fázi jsou aplikovatelné také na výpočet gradientu na vzorku. Zde je ale rozdíl, že vzorky už nutně neleží v bodech mřížky ale libovolně v mezi nimi. Při použití některé metody na výpočet gradientu se chybějící hodnota dopočte pomocí použitého rekonstrukčního filtru.

Ne ojedinělým případem je, že rekonstrukce se nepočítá po jednotkové ale jiné velikosti kroku. Pro případ centrálních diferencí je pak potřeba zapsané vztahy upravit tak, aby odpovídaly proměnlivé délce kroku. V tomto případě při výpočtu první derivace pomocí Taylorova polynomu bude potřeba uvažovat proměnné h. Vztahy pro aproximaci hodnoty $x \pm h$ jsou následující:

$$f(x+h) = f(x) + f'(x) \cdot h + f''(x) \cdot h^2/2 + o(h^3)$$
$$f(x-h) = f(x) - f'(x) \cdot h + f''(x) \cdot h^2/2 + o(h^3)$$

Stejnou úpravou pak dostaneme vztah pro výpočet první derivace:

$$f'(x) = \frac{f(x-1) + f(x+1)}{2h} + o(h^2)$$

Při úpravě rovnice je celá rovnice dělena h, proto řád aproximační chyby klesne o jeden stupeň. Aplikací tohoto vztahu lze pak aproximovat centrální diferenci obdobně jako v případě předpočítaného gradientu.

$$\nabla f(x,y,z) \approx \frac{1}{2h}\begin{pmatrix} f(x+h,y,z) - f(x-h,y,z) \\ f(x,y+h,z) - f(x,y-h,z) \\ f(x,y,z+h) - f(x,y,z-h) \end{pmatrix}$$

Metody konvoluce s větším jádrem lze také použít. Zde nastává problém, kdy je třeba uvažovat, jak moc zpřesněním výpočtu gradientu získá výsledný obraz kvalitu. Je potřeba uvažovat, že pro konvoluční jádra 3x3x3 je potřeba vzít v úvahu 27 vzorků pro výpočet jednoho gradientu, po 5x5x5 jádro konvoluce 125 vzorků, zatímco centrální diference vyžaduje vzorků pouze 6.

Před-počítání gradientu lze provést s velkou přesností a při vykreslování dat nebude náročnost jeho výpočtu hrát roli. Je ale potřeba uvažovat, že všechna data budou muset být

nahrána současně s primárními daty. Pro velké data sety toto řešení může být neúnosné z hlediska využité paměti.

Se stále více rostoucím výkonem počítačů a grafických karet se počítání gradientu na vzorku zdá jako příhodné řešení, které ušetří paměť na úkor náročnějšího výpočtu. Nicméně použití každého z těchto přístupů záleží až na konkrétní situaci. [13]

5.3.1.3 Použití osvětlovacích modelů

Jakmile už je jednou k dispozici gradient, lze na data aplikovat nějaký osvětlovací model (BRDF) použitelný pro polygonální modely. Při osvětlení volumetrických modelů oproti polygonálním, je důležitá také informace o velikosti gradientu. Jak již bylo zmíněno, homogenní oblasti dat izoplochy nemají. Gradient je i v tomto případě spočítán. V ideálním případě je spočítán jako vektor $(0,0,0)$. Homogenita oblastí však nemusí být nutně úplná. Menší sotva pozorovatelné rozdíly v intenzitách zapříčiní, že gradient už nebude nulový vektor. Šum vzniklý snímáním dat není žádnou výjimkou. Právě tento šum dokáže tyto drobné rozdíly vytvořit. Při aplikaci osvětlovacího modelu by s gradientem bylo počítáno tak, jako by se světlo odráželo uprostřed této téměř plně homogenní oblasti. Optický model tuto možnost nevylučuje. Metoda lokálního osvětlení má zjednodušující předpoklad, že světlo se odráží pouze od izoploch materiálu. Možnost odrazu světla v homogenní oblasti se díky tomu vylučuje. Proto je do počítání uvažována také délku vektoru gradientu. S jeho rostoucí délkou roste také jeho „důležitost“. Gradient vypočtený pro daný bod, jehož délka bude nulová nebo se bude blížit nule, bude mít minimální vliv na osvětlení dané oblasti.

Osvětlovací modely spadají do oblasti BRDF. Modelů existuje celá řada. Příklad známého modelu pro aplikaci BRDF je Blinn-Phongův osvětlovací model. Výsledné osvětlení je součtem tří složek: zrcadlová složka I_z, difúzní složka I_d a ambientní složka I_a.

$$I = I_a + I_d + I_z$$

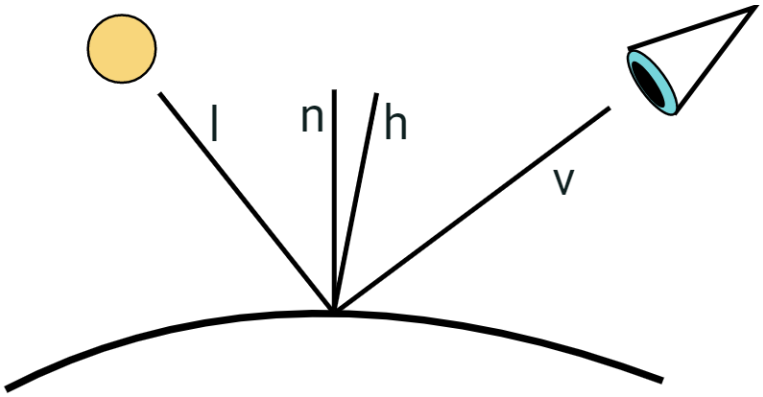
Zrcadlová složka počítá chování odrazu světla pro povrchy s vysokou mírou odrazivosti. Pro materiály jako leštěný kov se bude světlo odrážet ve velké míře ve stejném úhlu, v jakém dopadlo. Tato složka je závislá na normále povrchu, směru příchozího světla k povrchu, na směrovém vektoru k pozorovateli a na vektoru ve směru odrazu. Difúzní složka odpovídá Lambertově odrazivosti, která říká, že světlo dopadající na povrch tělesa se odrazí se stejnou pravděpodobností do všech směrů. Pro difúzní složku je podstatná normála povrchu a směrový vektor k pozorovateli. Ambientní složka modelu je v modelu pouze pro dobrý vizuální dojem. Může být zadaná i konstantně pro celou scénu. V reálném světě dochází k vícenásobným odrazům světla ale ne v případě lokálního osvětlovacího modelu. Aby pak regiony, které jsou odvrácené od zdroje světla, nezůstaly kompletně černé, což působí velmi

nerealisticky, byla zavedena ambientní složka, která se snaží vícenásobný rozptyl nahradit. Výpočet jednotlivých složek je dán následujícími rovnicemi.

$$I_a = k_a \cdot M_a \cdot I_s$$
$$I_d = k_d \cdot M_d \cdot I_s \cdot cos' \gamma$$
$$I_z = k_z \cdot M_z \cdot I_s \cdot (cos' \omega)^n$$

$\mathrm{I_a}$	...	Ambientní složka	M_d	...	Difúzní koeficient materiálu
I_d	...	Difúzní složka	M_z	...	Zrcadlový koeficient materiálu
I_z	...	Zrcadlová složka	I_s	...	Intenzita světelného zdroje
k_a	...	Ambientní koeficient	γ	...	Úhel mezi směrovým vektorem
k_d	...	Difúzní koeficient			k pozorovateli a normálou povrchu
k_z	...	Zrcadlový koeficient	n	...	Ostrost zrcadlového odrazu
M_a	...	Ambientní koeficient materiálu	ω	...	Úhel mezi half vektorem a normálou povrchu

Half vektor zjednodušuje výpočet zrcadlové složky. Phongův původní osvětlovací model předpokládá výpočet zrcadlové složky stejným vztahem, ale úhel ω odpovídá úhlu mezi vektorem odraženého světla a směrem k pozorovateli. V Blinn-Phongově modelu je úhel ω úhlem mezi normálou povrchu a half vektorem – „na půl cesty“ mezi vektorem k pozorovateli a vektorem ke zdroji světla. Apostrof u funkce cos značí, že funkce má upravený funkční předpis tak, že namísto záporných hodnot vrací 0. Vektory modelu jsou znázorněny na obrázku 11.



Obrázek 11 – Zobrazení vektorů pro Blinn-Phongů osvětlovací model. l je vektor ke světlu, n je normála, h je half vektor a v je vektor k pozorovateli od osvětlovaného bodu.

Koeficienty k_a, k_d a k_z jsou globální koeficienty. Často jsou společné pro celou scénu. Koeficienty M_a, M_d a M_z jsou koeficienty materiálu. Je běžnou praxí, že globální koeficienty a koeficienty materiálu se spojí do jednoho koeficientu společného pro celou scénu nebo těleso. Přechodová funkce nemusí navracet pouze optické vlastnosti zmiňované v předchozích

kapitolách, to znamená pouze barvu a průhlednost. Jejím výstupem mohou být i další vlastnosti materiálu jako jsou právě koeficienty M_a, M_d a M_z. Potom má význam oddělit globální koeficienty od koeficientů materiálu.

Jedním z možných přístupů, jak aplikovat velikost gradientu do výsledného osvětlení, je započítat osvětlení pouze pro body, jejichž velikost gradientu přesáhne danou mez. Velikost gradientu je závislá na rozdílu v intenzitách okolí. V případě, že rozdíl v intenzitách bude malý, velikost gradientu bude také malá, vzorek neleží na významné hraně oblastí a nebude osvětlen. [13]

5.3.2 Globální osvětlovací model

Lokální osvětlovací model je vhodný pro určité druhy dat, pro které podává dostatečné výsledky. Pro další skupinu dat ale dostatečný není. Existuje celá řada dat, jako jsou atmosférické mraky, kouř nebo vosk, kůže a další průhledné materiály, kde rozptyl světla nebo průchod světla skrz materiál dominuje vizuálnímu vjemu a pro tato data nedokáže lokální osvětlovací model poskytnout uspokojivý vizuální vjem. Stejně tak nepodává dostatečný výsledek pro tělesa, kde vržené stíny mají významný vliv na vizuální vjem.

Na globální osvětlovací modely lze z hlediska jejich zařazení do optických modelů (viz. kapitola 2.1) nahlížet několika způsoby. Globální osvětlovací modely lze zařadit do kategorie jednoduchého rozptylu světla i do vícenásobného rozptylu světla. Zařazení do příslušné kategorie lze provést pro konkrétní metodu vykreslování. I když vícenásobný rozptyl nebo výpočet rozptylu založený na fyzikálním základu bude podávat korektní výsledky odpovídající realitě, nejsou tyto podmínky nutností. Naopak častým případem je právě použití metod, které nejsou založeny na fyzikálním základy nebo heuristické metody (viz. ambientní složka osvětlení u lokálního osvětlovacího modelu). Ty jsou výpočetně mnohem méně náročné než fyzikálně korektní modely, ale výsledky, které podávají, mohou být pro pozorujícího člověka dostatečně věrohodné. Rozdíl mezi jednotlivými modely lze pozorovat na obrázcích 30-32 ve výsledcích.

Výpočet globálního osvětlení je dán tímto vztahem:

$$I(x,\omega) = \int_{\Omega} p(x,\omega,\omega') \cdot I(x,\omega') d\omega'$$

Veličiny odpovídají dříve zmíněným v předchozích kapitolách. Aplikace globálního osvětlovací modelu započítává náročnější jevy světla jako rozptyl. Výsledky vykreslování globálním osvětlovacím modelem působí ve většině případů lepším vizuálním dojmem ovšem na úkor výpočetní rychlosti. Metodou hrubé síly lze trasovat paprsek pohledu směrem od pozorovatele a pro každý vzorek na tomto paprsku trasovat další paprsek ke zdroji světla. Náročnost tohoto přístupu je ale $O(n \cdot m) \equiv O(n^2)$, kde n je počet vzorků primárních paprsků a m je počet vzorků pro sekundární paprsky. Výpočet tímto způsobem je přímou aplikací optického modelu jednoduchého rozptylu. Nejlepším výsledkem tohoto přístupu jsou tvrdé stíny. Výpočet lze dále rozšiřovat o další rozptyly světla na sekundárních paprscích. Náročnost

tohoto postupu roste geometrickou řadou. Jak lze vidět, tento výpočet je značně neefektivní. Navíc některé stíny se překrývají, a přesto se počítají pro každý vzorek znovu.

Jedním z přístupů, který vypočítá intenzitu dopadajícího světla přímo pomocí zadaného vztahu, trasování paprsku metodou Monte Carlo. Tato metoda není vhodná pro vykreslování v reálném čase, ale její aplikací lze získat přesné výsledky. I přes to autoři práce [19] využili tuto metodu pro vykreslování volumetrických dat v reálném čase, nicméně kvůli její náročnosti je potřeba využít různých zjednodušení. Toho bylo docíleno tím, že těleso je nejprve vykresleno bez odpovídajícího osvětlení a jeho výpočet se zpřesňuje a vykresluje s časem, pokud se nemění orientace tělesa nebo pozice pozorovatele. V této práci první výsledek vykreslování je vidět téměř hned, vykreslení finálního výsledku trvalo autorům 10 sekund.

Pro exploraci dat v reálném čase je třeba využít účinnější metody vykreslování. Jedná se o kompromis mezi kvalitou a rychlostí. Jednou z metod spadající částečně do globálního osvětlení je ambient occlusion. [13]

5.4 Ambient occlusion

Metoda ambient occlusion (AO) lze částečně zařadit do metod globálního osvětlovacího modelu, nicméně v této práci je zařazena do vlastní kapitoly. Aplikace metody pro objemová data vychází z původní metody pro povrchové modely. Proto je nejdříve popsána varianta pro povrchová data a následně rozšířena pro aplikaci na objemové modely.

5.4.1 Ambient occlusion pro povrchové modely

Cílem metody ambient occlusion, překládané jako zastínění okolím, je spočítat intenzitu dopadajícího světla do požadovaného bodu ve scéně. Výpočet odraženého světla směrem k pozorovateli při výpočtu globálního osvětlení je dán následujícím vztahem:

$$I(x,\omega) = \int_{\Omega/2} p(x,\omega',\omega) \cdot i(x,\omega') d\omega'$$

kde $I(x,\omega)$ je požadovaná intenzita světla v bodě x ve výsledném směru ω. $\Omega/2$ je symbolické označení pro povrch polokoule. Polokoule je centrovaná kolem normály povrchu n_x v bodě x. $i(x,\omega')$ je intenzita příchozího světla do bodu x z příchozího směru ω'. Právě tato složka je největší problém na výpočet, protože je daná rekurzivním vztahem, který započítává veškeré odražené a emitované světlo v celé scéně. Pro interaktivní vykreslování je tento výpočet stále příliš náročný a jeho zjednodušení na výpočet bez osvětlení a stínů vypadá nerealisticky a ploše, podobně jako počítání bez vícenásobného rozptylu světla v objemových datech. Namísto zjišťování intenzity světla příchozích paprsků, která může být výsledkem mnohonásobného odrazu paprsku přes velký počet těles ve scéně, se intenzita příchozího světla nezjišťuje vůbec. Když světlo ve scéně narazí na těleso, je odraženo a rozptýleno do několika směrů a tím přispává k dalšímu osvětlení ostatních objektů. Toto odražené světlo se nazývá ambientní. Metoda ambient occlusion pracuje právě s ambientní složkou světla a

vytváří měkké stíny tím, že ztmavuje body (povrchy) tělesa podle toho, jak jsou viditelné z blízkého okolí. Viditelností z blízkého okolí se nezjišťuje, jak je těleso viditelné pro zdroje světla, ale jaká část jeho blízkého okolí obsahuje nějaké jiné těleso a jaká část jeho okolí je prázdná. Předpokládá se, že tělesa s dalšími tělesy v blízkém okolí budou méně vystavena odraženému (ambientnímu) světlu a tím pádem méně osvětlené. Rovnice pro řešení AO je daná vztahem:

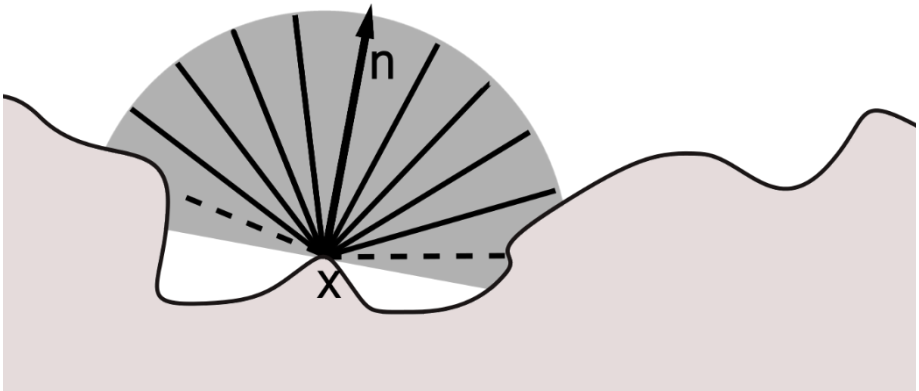
$$I(x,\omega) = \frac{1}{V}\int_{\Omega/2} v(x,\omega') \cdot (n \cdot \omega') d\omega'$$

pro který $v(x,\omega')$ je funkce vracející binární hodnotu v závislosti na tom, jestli paprsek z bodu x ve směru ω' je nebo není zastíněn. $(n \cdot \omega')$ je skalární součin normály n a směru vyslaného paprsku. Koeficient V je v rovnici z nutnosti normalizace, aby se výsledná hodnota pohybovala v intervalu $\langle 0,\ 1\rangle$. Formálně lze V zapsat vztahem:

$$V = \int_{\Omega/2} 1 d\omega'$$

Pro potřeby výpočtů je potřeba integrál diskretizovat na sumu funkcí $v(x,\omega')$, kde směry ω' budou zvoleny tak, aby dostatečně přesně aproximovaly všechny směry přes povrch polokoule a parametr V je roven počtu vyslaných paprsků.

Ukazuje se, že funkce $v(x,\omega')$ lze definovat i lépe. Její výsledek nemusí být pouze hodnota 0 nebo 1 (je v cestě těleso / není v cestě těleso), ale může vracet libovolnou hodnotu z intervalu $\langle 0,\ 1\rangle$. Výsledek této funkce lze pak interpretovat stylem: jak blízko blokující těleso v daném směru leží. Pro praktické použití tato funkce je definována jako monotónně rostoucí z hodnoty 0 do maximální vzdálenosti.



Obrázek 12 – Na obrázku je znázorněn bod povrchu x, pro který se počítá ambient occlusion. Normála povrchu n v bodě x určuje orientaci polokoule, která určuje blízké okolí bodu. Z obrázku lze vidět, že daný bod by byl po výpočtu částečně zastíněn. Jen 2 z 11 (jeden paprsek je shodný s normálou) vyslaných paprsků mají v blízkém okolí zastiňující těleso.

Výpočet viditelnosti pomocí diskretizace integrálu je znázorněno na obrázku 12. Jedná se pak o zjednodušení globálního modelu. Existuje více možností, jak vypočítat viditelnost bodu metodou AO. [22,32]

Ambient occlusion pomocí trasování paprsku (ray traced ambient occlusion – RTAO) je metoda, která odpovídá nejpřesněji zadanému vztahu pro jeho výpočet. Výpočet pomocí trasování paprsku je výpočetně náročný. Rovnice obsahuje integrál přes povrch polokoule, který se aproximuje použitím dostatečného množství paprsku směřující do co největšího množství směrů. Rostoucí počet paprsků zvyšuje přesnost a současně zvyšuje výpočetní náročnost. [34]

Metoda ambient occlusion v prostoru obrazu (screen space ambient occlusion – SSAO) počítá zastínění až ve výsledném obrazu. Předpokladem je, že pouze finálně vykreslená tělesa mohou mít vliv na osvětlení. Autor se tím vyhne náročnému výpočtu viditelnosti bodu v trojdimenzionálním prostoru scény. Je toho docíleno vzorkováním hloubkových map na grafické kartě. Výsledky tohoto přístupu nejsou fyzikálně přesné, nejsou započteny všechny objekty, které by mohly bránit viditelnosti objektu. Jednoznačnou výhodou tohoto přístupu je konstantní čas a konstantní využití paměti. [26]

Ambient occlusion podle horizontu (horizon-based ambient occlusion – HBAO) probíhá počítáním maximálního úhlu nad kterým může světlo svítit. Metoda také probíhá v prostoru obrazu. Tento přístup je relativně efektivní, ale i přes to ne plně optimální. Na nalezení maximálního horizontu je potřeba započítat relativně hodně vzorků z hloubkové mapy. [4]

Metody AO založené na jeho počítání v prostoru obrazu trpí nedostatky, převážně těmito:

- tmavé oblasti okolo objektů a nedostatek zastínění za objekty v popředí,
- nestálé výsledky okolo okrajů obrazu,
- lokálnost přístupu – pouze některé objekty v malém blízkém okolí přispívají do zastínění,
- rozmazanost – na výsledek je potřeba aplikovat rozostřovací filtr, protože počítat kompletní řešení pro každý pixel by bylo příliš náročné.

Voxel ambient occlusion (VXAO) je metoda, která viditelnost objektu počítá v prostoru scény. Vychází z metody VXGI (Voxel Global Illumination). VXGI má za cíl počítat globální osvětlení scény a toho dosahuje pomocí voxelizace scény – to znamená, celá scéna je převedena do objemové datové struktury (volumetrická textura), nad kterou se provádí výpočty osvětlení. Oproti tradičním přístupům je VXGI efektivnější právě díky využití voxelů, které jsou uniformní a s tím souvisí mnohem snadnější trasovaní paprsku. Techniku VXGI používají například autoři práce [33]. Namísto počítání celého osvětlení, VXAO nepočítá osvětlení ale pouze zastínění daného bodu. Díky vynechání osvětlení dosahuje VXAO 2-10x větší výpočetní rychlosti než VXGI v závislosti na nastavení vykreslování a scény. Metoda je 3-4x pomalejší než už zmiňovaná metoda HBAO, ale její výsledky jsou v porovnání výrazně lepší a netrpí zmíněnými nedostatky. Tento algoritmus probíhá ve třech základních krocích:

1. Voxelizace scény. Tento krok převede scénu do objemové textury. Krok je závislý na počtu polygonů ve scéně, jejich velikosti a počtu buněk potřebných k jejich vykreslení, ale jeho náročnost je pořád dostatečně únosná, že ho lze na moderních grafických kartách (autor uvádí NVidia GeForce GTX 980) počítat pro každý snímek i pro několik milionů trojúhelníků v čase kolem 3-5ms.
2. Post-processing voxelů zahrnuje filtrování, škálování voxelů a čištění dat. Výkon druhého kroku závisí na celkovém počtu voxelů vygenerovaných v první fázi. Typický čas druhého kroku je 0,5-1,5ms.
3. Trasování z prostoru obrazu. Jeho výkon je pak závislý na rozlišení finálního obrazu.

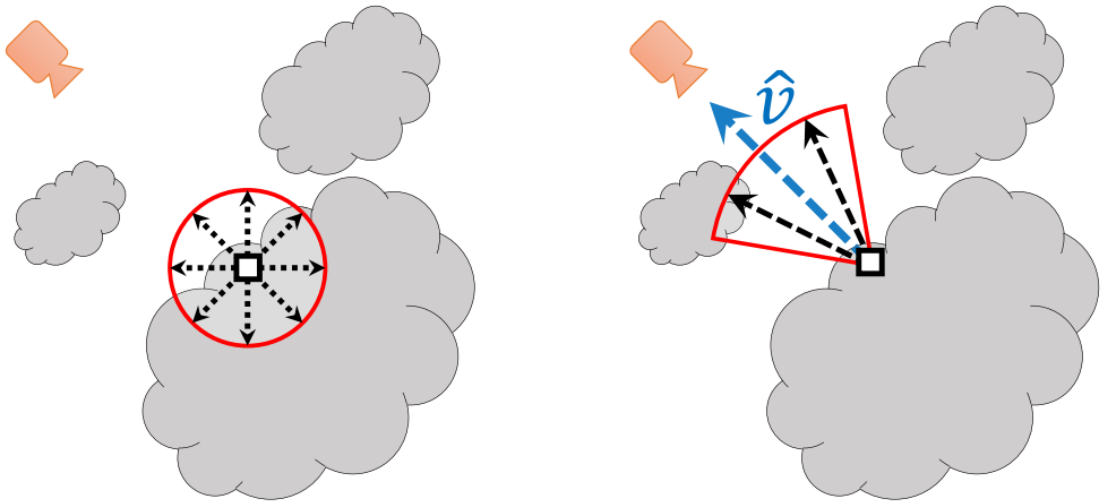
Algoritmus si poradí i dynamickými scénami. Všechny kroky algoritmu se mohou vypočítat pro každý snímek v reálném čase. Náročnost na paměť je také únosná. Typicky se využití paměti touto texturou pohybuje v rozmezí od 6 do 100 MB. [2]

5.4.2 Ambient Occlusion pro objemové modely

Metoda VXAO má společné kroky s metodami AO pro objemové modely. V momentě, kdy jsou všechna data polygonálních modelů připravena a uložena do trojrozměrného datového souboru, práce s tímto souborem již může využívat objemové metody.

V zásadě lze AO objemových modelů rozdělit na dvě kategorie podle toho, jaké okolí se uvažuje pro výpočet útlumu světla. První z kategorií uvažuje plné kruhové okolí daného bodu. Znamená to tedy, že rozptyl světla uvažuje jak všechny body tělesa ležící blíže ke světlu než daný bod, tak také všechny body ležící dále, než daný bod. Výpočet při uvažování tohoto přístupu vychází z výpočet AO pro povrchové modely. Tento výpočet je ale potřeba upravit. Pro povrchové modely stačí integrovat pro směry polokoule nad daným bodem centrovanou kolem normály v daném bodě. Pro objemová data tento integrál bude započítávat všechny směry kulové plochy. Dále pro objemová data platí, že pro ně neexistuje normála, ale pouze gradient. Ten lze považovat za normálu izoplochy procházející daným bodem, ale jeho velikost značně určuje, jak moc ostrá hrana se nachází v daném bodě. Výpočet integrálu je složitý, a proto se běžně pro tuto kategorii využívá výpočet lokálního zastínění v okolí bodu, což je rychlejší přístup na úkor přesnosti – například zjištěním intenzit vybraných vzorků v okolí bodu. [7]

Druhá kategorie uvažuje pouze okolí tvaru kužele s vrcholem v daném bodě a s jeho orientací směrem k pozorovateli. Namísto výpočtu integrálu pro všechny směry pro povrch koule, při výpočtu integrálu pouze pro jejich část ležící v kuželu se výpočet značně zjednoduší. [8]

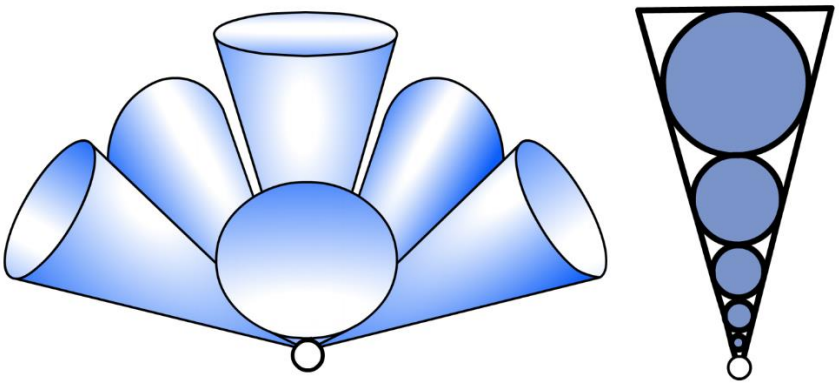


Obrázek 13 – Ukázka dvou kategorií AO pro objemová data. Vlevo: AO počítané pro lokální okolí, které leží uvnitř hraniční koule. Vpravo: směrové AO. Do výsledného útlumu světla se započítají pouze body nebo paprsky (podle zvolené metody) ležící uvnitř kužele směřující k pozorovateli. Zdroj: [8]

5.5 Trasování kužele

Trasování kužele je metoda, která umožňuje účinně získat množství zastíněného světla do daného bodu v objemu. Využití trasování kužele je značné – výpočty globálního osvětlení v objemových datech, výpočet VXAO (viz kapitola Ambient occlusion) nebo výpočet AO v objemových datech. Přímočaré řešení této metody je implementace způsobem metod Monte Carlo. Z vrcholu kužele bude vysláno značné množství paprsků ve směru kužele. Pro každý z paprsků bude zjišťován útlum světla a výsledný útlum bude dán funkcí všech zastínění – například vážený průměr podle kosinu úhlu mezi výškou kužele a směrem paprsku.

Autoři práce [11] využívají trasování kužele k počítání ambient occlusion pro povrchové modely s voxelizací scény. Scéna je převedena do reprezentace, která ukládá oblasti scény do binární informace – oblast obsahuje nebo neobsahuje geometrii. Aby byla zachována korektnost výpočtu AO pro povrchové modely, je v práci zjišťováno zastínění bodu okolní geometrií na hemisféře kolem bodu. Aplikací několika kuželů s daným úhlem nahrazuje trasování paprsků. Výsledné zastínění bodu je pak součtem zastínění jednotlivých kuželů. Při výpočtu je každý kužel nahrazen skupinou koulí tak, že v rámci kužele na sebe navazují. Znázornění nahrazení hemisféry kolem bodu skupinou kuželů a nahrazení kužele skupinou koulí je na obrázku 14.



Obrázek 14 – Vlevo: Pro výpočet zastínění bodu lze hemisféru nad bodem nahradit skupinou kuželů. Výpočet zastínění je pak řešením výpočtu zastínění těchto kuželů. Vpravo: Výpočet zastínění jednotlivého kužele je možné převést na výpočet zastínění několika koulí, kterými lze vhodně tento kužel nahradit. Zdroj: [11]

Autoři práce [11] používají toto zastínění na výpočet AO v prostoru obrazu. Dostupné informace jsou uložené ve voxelizované reprezentaci scény, v jejich případě ve vhodně upravené dvojrozměrné textuře. Výpočet zastínění jednotlivé koule je pak dán jako poměr zastíněných vzhledem k celkovému počtu vzorků v oblasti koule. V prostoru obrazu to znamená vhodně přepočítat souřadnice do textury a z nich vyčíst zastínění. Samotné skládání zastínění koulí podléhá kompozitnímu schématu.

Pro snadnější výpočet zastínění jednotlivých koulí by bylo vhodné, kdyby informace o zastínění procházejícího světla byla již dána pro každý střed koule. Data musí být vhodně upravena, aby toto kritérium splňovala. Jedním z řešení je na data aplikovat Gaussův filtr [8].

6 Zobrazení objemových dat

Zobrazení objemových dat probíhá pomocí metody raycastingu popsané v kapitole 5.1. Samotná metoda raycastingu však nevystačí na aplikaci složitějšího osvětlovací modelu. Už v předchozích kapitolách bylo zmíněno, že raycasting je přímou aplikací emisně-absorpčního modelu. Cílem práce je zobrazení objemových dat s využitím techniky, která využívá komplexnější osvětlovací model – jednoduchý nebo vícenásobný rozptyl světla.

V této práci jsou implementovány dva rozdílné přístupy, které rozšiřují raycasting. První z přístupů, který vychází z práce autorů [8], podává výsledky založené na reálném fyzikálním základu. Druhý autorský přístup si klade za cíl uspořit výpočetní výkon předpočítáním stínů. Předpokladem druhého přístupu je, že výpočet stínů dopadajících od zdroje světla k počítanému bodu je dán funkcí všech předchozích bodů ve směru ke zdroji světla s určitým rozptylem. Oba tyto přístupy, stejně jako implementace využitých metod jsou popsány v dalších podkapitolách.

Veškerá implementace v této práci je vytvořena v jednotném prostředí. Zdrojový kód využívaný pro spuštění na procesoru je napsán v jazyce Java. Programování grafické karty je vytvořeno za použití knihovny OpenGL, jejíž funkčnost v jazyce Java je zajištěno knihovnou LWJGL. Shaderové programy jsou psané v programovacím jazyce GLSL. Odkazy na specifikace využitých jazyků a knihovnu jsou umístěny za zdroje v kapitole 9.

6.1 Implementace raycastingu

Metoda raycastingu obecně trasuje paprsky od průmětny pozorovatele do prostoru scény pro všechny pixely průmětny. Vyhodnocování a trasování paprsku je zbytečné pro body v prostoru, které leží mezi obálkou tělesa a pozicí pozorovatele, nebo pro body, které leží na paprsku, který obálku ani neprotíná. S využitím standardních souřadnic do trojrozměrné textury uvw, interval $\langle 0,1 \rangle$ zahrnuje veškeré hodnoty textury. Funkce vracející hodnotu textury vrací hodnotu i pokud její parametr leží mimo daný interval. Hodnoty pro parametry funkce mimo tento interval budou dopočítány podle zvolené metody. Pro správné fungování je funkce je nastavena tak, aby pro tyto parametry vracela nulovou hodnotu, tzn. prázdné body. Nicméně tyto hodnoty je při počítání barevného příspěvku zbytečné procházet. Proto je potřeba použít metodu, která určí, pro které paprsky a v jakém rozsahu má trasování paprsku smysl provádět.

Metoda je založena na zobrazení obálky ve tvaru kvádru ohraničující objemovou texturu. Všechny trasované paprsky procházející materiálem musí současně protnout obálku tělesa a to dvakrát. Jednou při vstupu a jednou při výstupu. Vykreslováním přední strany obálky zůstanou ve vykreslovacím procesu pouze ty pixely, pro které má smysl paprsek trasovat.

Grafická karta je uzpůsobena pro vykreslování polygonálních modelů, nikoliv objemových. Zde ale s výhodou využijeme povrchový model obálky tělesa, uvnitř kterého raycasting probíhá. Při jeho implementaci na grafické kartě je iterační cyklus algoritmicky

zpracován ve fragment shaderu. Pro každý pixel zobrazeného obrazu je jedenkrát spuštěn trasovací cyklus. Při aplikaci prostorových transformací (modelovací, pohledová a projekční) na geometrii obálky tělesa a se zapnutým ořezáváním polygonů odvrácených od pozorovatele, se program ve fragment shaderu spustí pouze pro ty pixely, které vzniknou rasterizací přední strany obálky. Tím je zajištěno, že paprsky světla neprotínající materiál nejsou do výpočtů zahrnuty.

Stále zbývá vyřešit problém, aby trasování paprsku nezačalo v průmětně pozorovatele. V ideálním případě trasování paprsku začne na hraně obálky tělesa, a tím bude zajištěno, že iterační cyklus nebude iterovat přes prázdný prostor mimo těleso. Jak již bylo zmíněno v předchozím kroku, každý pixel průmětny, pro který bude trasování probíhat, už bude pomocí prostorových transformací ve vertex shaderu transformován. Vzniklé polygony se rasterizují na dané požadované pixely a pro každý pixel lze současně zaslat do fragment shaderu pozici bodu prostoru, kde má trasování začít. Do výstupu vertex shaderu se přidá další parametr určující tuto pozici. Celý raycasting může probíhat v soustavě souřadnic světa nebo v soustavě souřadnic pozorovatele. V obou těchto případech jsou modely těles umístěny do prostoru a současně nejsou deformovány projektivní transformací. V implementaci je použita varianta trasování v soustavě souřadnic světa.

Vrcholy obálky jsou transformovány dvakrát a každá transformace má vlastní výstup do vykreslovacího řetězce. První transformace zůstává modelovací, pohledovou a projekční dohromady. Druhá transformace je pouze modelovací transformací tělesa, kdy je každý vrchol transformován do soustavy souřadnic světa. Výstup první transformace je zpracován zobrazovacím řetězcem, zatímco výstup druhé transformace je lineárně interpolován a předán do fragment shaderu. Směr trasování paprsku je pak určen v soustavě souřadnic světa jako vektor od pozice pozorovatele směrem k pozici fragmentu. Vypočítaný směr trasování musí být v momentě, kdy protne obálku tělesa, transformován do soustavy souřadnic modelu.

```
#version 330
in vec3 inPosition; // vstupní pozice v prostoru modelu
out vec3 coordsMS; // výstupní souřadnice v prostor modelu
out vec3 coordsWS; // výstupní souřadnice v prostor světa
uniform mat4 model4; // modelová transformace
uniform mat4 MVP; // transformace modelová, pohledová a projekční
void main() {
    // výstup obálky tělesa do vykreslovacího řetězce
    gl_Position = MVP * vec4(inPosition, 1.0);
    // transformace do soustavy souřadnic světa
    coordsWS = (model4 * vec4(inPosition, 1.0)).xyz;
    // souřadnice do textury se nezmění - souřadnice v prostoru modelu
    coordsMS = inPosition;
}
```

Ukázka 1 – Zdrojový kód implementace raycastingu na vertex shaderu. gl_Position je odchozí pozice, která po průchodu zobrazovacím řetězcem zajistí rasterizaci pro pixely, pro které má smysl trasovat paprsek. Proměnné coordsMS a coordsWS jsou pozice, které se dále interpolují pro jednotlivé pixely. Každý sufix MS značí soustavu souřadnic modelu (model space) a WS světa (world space).

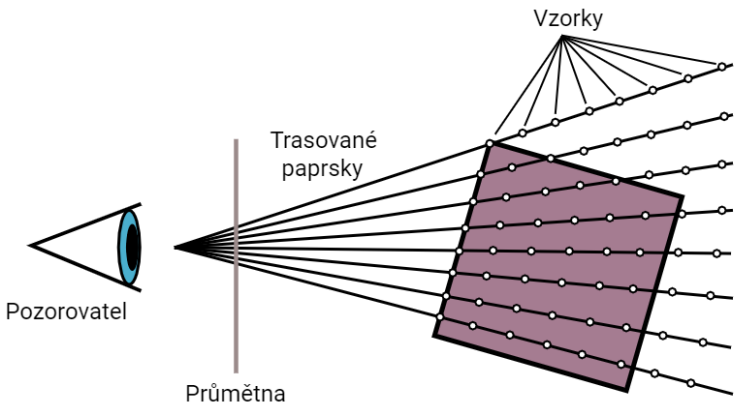
Další potřebný parametr je pozice vrcholů v soustavě souřadnic modelu. Vrcholy obálky odpovídají vrcholům jednotkové krychle s vrcholy $[0,0,0]$, $[0,0,1]$, $[0,1,0]$, atd. To zajišťuje, že souřadnice do textury uvw odpovídají souřadnicím vrcholů v soustavě souřadnic modelu. Tím jsou usnadněny výpočty souřadnic do textury pro jednotlivé vzorky při trasování paprsku. Implementace na vertex shaderu je zobrazena v ukázce 1 a ve fragment shaderu je zobrazena v ukázce 2. Zobrazení metody raycastingu je na obrázku 15.

```
#version 330
in vec3 coordsWS; // vstupní lineárně interpolovaná pozice
in vec3 coordsMS; // interpolovaná souřadnice do textury
out vec4 outColor; // výstupní barva

uniform sampler3D src; // uložená diskrétní data
uniform sampler2D preintegratedTransferFunction; // přechodová funkce
uniform vec3 eye; // pozice pozorovatele
uniform mat3 iModel3; // inverzní modelová transformace 3x3
uniform float voxelSize; // velikost voxelu
uniform float stepSize; // délka jednoho kroku cyklu

void main() {
    // výpočet směrového vektoru v prostoru světa (world space - WS)
    vec3 viewDirWS = normalize(coordsWS-eye);
    // přepočet směrového vektoru do prostoru modelu (model space - MS)
    vec3 viewDirMS = (iModel3 * viewDirWS)*voxelSize*stepSize;
    //inicializace ukazatelů do předintegrované přechodové funkce
    vec2 ptfPointer = vec2(0, 0);
    // délka tělesové úhlopříčky = limit pro iteraci
    int maxIt = int(length(textureSize(src, 0))/stepSize+1);
    for (int index = 0; index < maxIt; index++){
        // výpočet souřadnice do textury
        vec3 posMS = coordsMS + index*viewDirMS;
        // čtení skalární hodnoty sb
        ptfPointer.y = texture(src, posMS).r;
        // aplikace před-integrované přechodové funkce
        vec4 color = texture(preintegratedTransferFunction, ptfPointer);
        // sb aktuálního kroku se rovná sf následujícího kroku
        ptfPointer.x = ptfPointer.y;
        // do výpočtu se započítají pouze neprůhledné vzorky
        if(color.a <= 0.01 )
            continue;
        // opacity correction - výpočet nové opravené barvy
        vec4 corrected = vec4(0,0,0,0);
        corrected.a   = 1-pow(1-color.a, stepSize);
        corrected.rgb = color.rgb*corrected.a/color.a;
        // výpočet výsledné barvy s asociovanými barvami
        outColor.rgb = outColor.rgb + (1-outColor.a)*corrected.rgb;
        outColor.a   = outColor.a   + (1-outColor.a)*corrected.a;
    }
}
```

Ukázka 2 – Zdrojový kód implementace raycastingu na fragment shaderu.



Obrázek 15 – Ukázka principu raycastingu. Z pozice pozorovatele se trasují paprsky protínající průmětnu. Pro každý paprsek se iteruje cyklus, který akumuluje barvu a opacitu. První bod iterace je bod ležící na hraně obálky tělesa a dále se iteruje podle dané délky kroku.

Takto ukázaná metoda raycastingu je funkční, ale ne optimální. Například z obrázku 15 lze vidět, že vzorky na paprsku se snímají i poté, co už paprsek opustil prostor obálky, ale iterační cyklus je nastaven na nejdelší možnou vzdálenost, kterou paprsek uvnitř obálky může urazit. Je žádoucí použít nějaké optimalizační metody, například metody zmíněné v kapitole 5.1.

6.1.1 Počítání v různých soustavách souřadnic

V ukázce raycastingu je vypočtený směrový vektor paprsku od pozorovatele transformován inverzní modelovou maticí 3x3. Je to čistě z praktického důvodu. Samotná transformace obsahuje veškeré informace potřebné k tomu, aby vektor v SSS (soustava souřadnic světa) byl převeden do SSM (soustava souřadnic modelu). Tím se zjednoduší další výpočty pro výpočet souřadnic do textury, které leží na osách xyz v intervalech $\langle 0,1 \rangle$.

Nicméně pokročilejší metody globálního osvětlení, stejně tak jako metody lokálního osvětlení, vyžadují práci v SSM i SSS. Metody osvětlovacích modelů počítají vektor ke světlu z bodu uvnitř objemu. Výjimkou nejsou ani metody implementované v této práci, viz kapitola 5. Jeden z možných přístupů je veškeré výpočty provádět v intuitivní SSS, kde není potřeba provádět transformace v mezikrocích, ale je potřeba provádět transformaci na vypočtený bod případně vektor, aby se mohla určit jejich poloha / směr v SSM. V této práci je zvolen jiný přístup. Již před samotných cyklem na fragment shaderu je směrový vektor směřující skrz objem transformován z SSS do SSM. Výpočet nové pozice bodu uvnitř objemu je přímočará záležitost vynásobení směrového vektoru iterátorem cyklu a přičtení k souřadnicím vstupního bodu v SSM (viz implementace ray castingu). Takto získaný bod přímo odpovídá souřadnicím bodu v trojrozměrné textuře. Výpočet vektoru z tohoto bodu ke zdroji světla probíhá v několika krocích.

V prvním kroku je bod uvnitř objemu v SSM transformován modelovou maticí 4x4 do SSS. Odečtením pozic dvou bodů v SSS je výsledkem vektor znovu v SSS. V této fázi probíhá

normalizace vektoru na jednotkovou délku. Následně je inverzní modelovou maticí 3x3 transformován do SSM. V SSM už normalizace neprobíhá. Díky tomu transformovaný vektor v SSM odpovídá jednotkové délce v SSS. Ve zdrojovém kódu implementace nesou různé proměnné sufix MS (model space – SSM) a WS (world space – SSS) značící příslušnost do dané soustavy souřadnic.

6.1.2 Aplikace optimalizačních metod

Využití optimalizačních metod je žádoucí jednak kvůli zvýšení výpočetního výkonu, ale také na zvýšení kvality výsledného obrazu. Jednotlivé metody se liší i náročností implementace. V následujících podkapitolách je postupně popsána implementace jednotlivých optimalizačních metod popsaných v kapitole 5.1 věnované právě optimalizaci vykreslování.

6.1.2.1 Aplikace early ray termination

Název metody lze přeložit jako předčasné ukončení trasování paprsku. Jak bylo popsáno dříve, ukončení trasování paprsku pro raycasting lze ve dvou případech – pokud se naakumulovaná opacita dostane na dostatečně vysokou hodnotu nebo pokud pozice vzorku na paprsku opustí obálku tělesa.

První z těchto případů je značně přímočarý. Pro každý krok cyklu postačí podmínka, která přeruší cyklus, pokud naakumulovaná hodnota překročí daný práh. Implementace je zobrazena v ukázce 3. Tato ukázka stejně jako veškeré další ukázky, pokud není napsáno jinak, vycházejí ze základního fungování metody raycastingu popsané v předchozí kapitole 6.1, z ukázky 2.

```
// ... hlavička
void main(){
    // ... příprava před cyklem
    for (int index = 0; index < maxIt; index++){
        // ...
        if(outColor.a > 0.99) // podmínka přerušení
            break;
    }
}
```

Ukázka 3 – První případ možného přerušení trasování paprsku, pokud naakumulovaná opacita překročí danou hranici.

Druhý z případů, kdy se trasování může přerušit, je, pokud pozice vzorku na paprsku leží mimo obálku (bounding box). Pohled na tento problém není jen jeden. Směrový vektor od pozorovatele směrem do tělesa je dán pozicí pozorovatele, která je zadaná v soustavě souřadnic, a bodem na přední stěně obálky, který byl interpolován a je také v SSS. Vypočtený vektor je také v SSS. Při znalosti velikosti obálky tělesa (těleso má rozdílnou velikost SSM a v SSS) není problém určit, zda vzorek leží uvnitř nebo ne. Druhý přístup, využívaný v implementaci této práce, je transformace vektoru inverzní modelovou maticí 3x3 a

převedení vektoru do SSM. Jedna transformace před samotným průběhem cyklu zjednoduší samotnou podmínku od obecných výpočtů vnitřku obecné obálky na zjištění pouze toho, zda aktuální všechny složky xyz vzorku leží v rozsahu $\langle 0,1 \rangle$. Výsledná implementace je v ukázce 4. Trasování paprsku začíná a probíhá v SSM a první bod má souřadnici právě takovou, že alespoň jedna složka je nulová nebo se rovná jedné a pohledový vektor uvnitř tělesa se transformuje tak, že směřuje vždy do tělesa. Právě díky tomu je bezpečně možné takovou podmínku aplikovat.

```
// ... hlavička
void main(){
    // ... příprava před cyklem
    for (int index = 0; index < maxIt; index++){
        vec3 coordinatesMS = coordsMS + index*viewDirMS;
        if( //podmínka přerušení
            coordinatesMS.x > 1 || coordinatesMS.x < 0 ||
            coordinatesMS.y > 1 || coordinatesMS.y < 0 ||
            coordinatesMS.z > 1 || coordinatesMS.z < 0
        ) break;
        // ...
    }
}
```

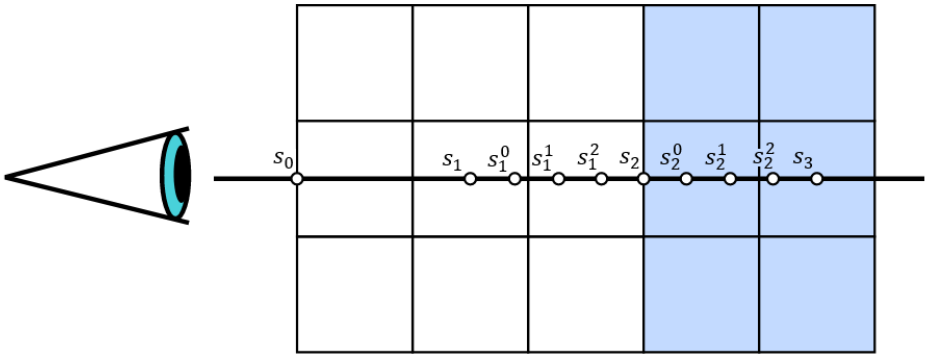
Ukázka 4 – Ukázka obsahuje druhou možnost k předčasnému přerušení cyklu. Aby bod ležel uprostřed tělesa, musí všechny z jeho souřadnic ležet v intervalu $\langle 0,1 \rangle$. Pokud jediná souřadnice leží mimo tento interval, znamená to, že vektor již opustil těleso a je bezpečné cyklus přerušit.

6.1.2.2 Aplikace adaptive samplingu

Bez adaptive samplingu pracuje cyklus v raycastingu v pravidelných krocích. Dokud iterátor trasovacího cyklu nemá dostatečně vysokou hornotu, aby nesplnil podmínku iterace nebo není ukončen cyklus jinak, vypočte se pro každý krok cyklu nová pozice vzorku jako vstupní pozice do objemu zvětšená o násobek iterátoru a směrového vektoru dovnitř objemu. Je žádoucí, aby vzorkovací frekvence byla minimálně 2x větší, než je maximální frekvence snímaného signálu. V praktickém použití to znamená, aby počet vzorku na jeden voxel byl větší než dva. Pro prázdné oblasti objemu, což jsou oblasti, kde přechodová funkce přiřadí nulový extinkční koeficient, tj. kde je nulová opacita, to znamená zbytečné vzorkování znovu prázdných míst bez efektu na výsledek obrazu. Stejně tak pro homogenní oblasti. Žádoucí je upravit vzorkovací frekvenci podle toho, zda je na aktuálním úseku paprsku objem homogenní nebo heterogenní. Jedno z existujících řešení je pomocí předpočítání *importance volume*, ale nebylo zde použito.

Namísto něj byl použit jiný přístup. Vzorkovací frekvence je nastavena na vyšší, než připouští vzorkovací teorém pro korektní rekonstrukci signálu. Filtrování objemových dat na grafické kartě je nastaveno na *tent* filtr (lineární interpolace). Dva následující vzorky budou dopočítány z jejich okolních voxelů díky tomuto filtrování. Pokud vzorkovací frekvence zůstane nižší než dvojnásobek velikosti strany voxelu, pak jsou všechny vzorky po směru paprsku započteny minimálně do jednoho vzorku tohoto paprsku s nenulovou vahou.

Dva následující vzorky tak obsahují příspěvky voxelů na úseku mezi nimi. Pokud by nějaký voxel započítaný do vzorku měl jinou intenzitu než ostatní započítané voxely, jeho odlišná hodnota se promítne. Je zřejmé, že pokud dva následující vzorky navracejí různou intenzitu, minimálně jeden z voxelů ležící na paprsku má odlišnou hodnotu a je potřeba na tomto úseku zvýšit úroveň vzorkování mezi těmito vzorky. Obrázek 16 zobrazuje, jak se tento přístup chová na hraně dvou prostředí.



Obrázek 16 – Princip fungování implementované metody. Vzorkovací frekvence je nižší, než aby dávala korektní výsledky. Body s_0, s_1, s_2 a s_3 jsou pozice primárních vzorků. Pokud je interpolací zjištěna odlišná intenzita na dvou po sobě jdoucích vzorcích, je potřeba mezi nimi aplikovat vyšší vzorkovací frekvenci. Vzorek s_2 leží na hraně dvou prostředí, interpolací je mu dopočtena jiná intenzita, než intenzita s_1, proto je nutné zvýšit frekvenci a mezi zahrnout do výpočtu nové vzorky s_1^0, s_1^1 a s_1^2. Stejně tak pro další úsek.

Přístup je implementován pomocí dvou do sebe vnořených cyklů. Vnější cyklus je stejný cyklus jako v ukázce raycastingu. Počet iterací vnořeného cyklu je zadán jako parametr programu, je nastavitelný uživatelem. Ve vnořeném cyklu probíhají veškeré výpočty raycastingu. Pokud jsou dvě následující intenzity stejné, vnořený cyklus proběhne pouze jedenkrát. V opačném případě je počet iterací roven zadanému parametru. Algoritmus je zobrazen v ukázce 5.

Podmínka určující, jestli je potřeba mezi dvěma následujícími vzorky zvyšovat vzorkovací frekvenci, je nastavena bez jakékoliv tolerance. Přechodová funkce zadaná uživatelem může být zcela libovolná. Potom i minimální změna v intenzitě zdrojového objemu může vyústit v radikální změnu optických vlastností. Například z plně průhledné do plně neprůhledné. Tato změna představuje vysokou frekvenci signálu. Pokud by na tuto změnu nebyla aplikována zvýšená vzorkovací frekvence, došlo by ke vzniku významných artefaktů v obraze. Důsledkem toho je také intolerance na jakýkoliv šum.

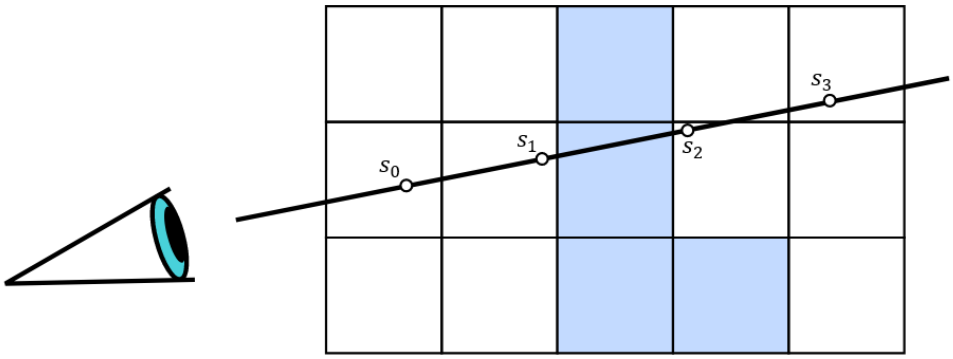
Druhým důvodem, proč je podmínka nastavena bez tolerance, je kvůli chybnému určení hranice dvou různých prostředí. Délka úseku mezi dvěma vzorky primárního paprsku Δs je libovolné číslo z intervalu $(0,2)$. Může nastat situace, kdy interpolované intenzity na dvou následujících vzorcích budou stejné, ale prostředí mezi nimi se změní, viz obrázek 17. Tento případ může reálně nastat při vykreslování s touto metodou, nicméně výsledky ukazují, že na testovaných datech tyto artefakty nebyly významné nebo byly úplně zanedbatelné.

```
// ... hlavička
uniform int adaptiveSamplingMultiplier; // násobek vzorkovací frekvence

void main(){
    // inicializace intenzity aktuálních a následujících vzorků
    vec2 intensities = vec2(0,0);
    // ...
    // primární cyklus
    for (int index = 0; index < maxIt; index++){
        // výpočet pozice následujícího vzorku
        posMS = coordsMS + viewDirMS*(index+1);
        // přiřazení intenzity následujícího vzorku paprsku
        intensities.y = textureLod(textureVol, posMS, LOD).r;
        // délka mezikroku jako násobek původní vzorkovací frekvence
        float innerCycleInc;
        // násobek původní vzorkovací frekvence
        int innerSamplingMultiplier;
        if(intensities.x == intensities.y){
            innerCycleInc = 1.0;
            innerSamplingMultiplier = 1;
        }else{
            innerSamplingMultiplier = adaptiveSamplingMultiplier;
            innerCycleInc = 1.0/adaptiveSamplingMultiplier;
        }

        // vnitřní cyklus
        for(int iIndex = 0; iIndex < innerSamplingMultiplier; iIndex ++){
            posMS = coordsMS + viewDirMS*(index + iIndex*innerCycleInc);
            // ...
            corrected.a = 1-pow(1-color.a, stepSize*innerCycleInc);
            // ...
        }
        intensities.x = intensities.y;
    }
}
```

Ukázka 5 – Implementace adaptive samplingu pomocí vnoření dvou cyklů. V každém kroku primárního cyklu je potřeba zjistit intenzitu následujícího vzorku. Pokud se intenzity dvou následujících kroků liší, je potřeba ve vnitřním cyklu zvýšit frekvenci vzorkování.



Obrázek 17 – Vzorky paprsku s_1 a s_2 leží ve stejné vzdálenosti od středu světle modré oblasti. Interpolací dopočtené hodnoty jsou stejné, a proto nedojde ke zvýšení frekvence vzorkování na úseku paprsku mezi těmito body, i když je patrné, že v tomto úseku je vzorkování nedostatečné.

6.1.3 Aplikace lokálního osvětlovacího modelu

Lokální osvětlovací model se v implementaci počítá pro každý vzorek při samotném vykreslování. Z důvodu výpočetního výkonu je pro osvětlení použit výpočet centrálních diferencí. Důvodem je jednoznačně vyšší rychlost výpočtů oproti větším konvolučním maskám i za cenu snížené přesnosti.

Výpočet centrálních diferencí popsaný v kapitole 5.3.1.2 probíhá v SSS. Celý implementovaný výpočet probíhá v SSM. V této soustavě odpovídají okolní body přímo souřadnicím do textury a je ušetřen výkon za tento přepočet. Aby však mohly být korektně určeny souřadnice vzorků, odkud se budou intenzity načítat, je potřeba vhodným způsobem centrální diference převést do SSM. Pro všechny tři souřadnicové osy stačí koeficient h vhodně škálovat.

Souřadnice do textury stejně jako souřadnice obálky objemu jsou tyto: (u, v, w), kde $u, v, w \in \{0,1\}$. Modelovací transformace obálky tělesa je složena ze dvou transformací: centrování obálky jejím posunutím na střed – posunutí všech bodů o vektor $\left(-\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}\right)$ a škálování obálky vektorem $(w \cdot v, h \cdot v, d \cdot v)$, kde w, h, d jsou rozměry objemu – šířka, výška a hloubka, a v je konstanta – velikost strany voxelu v SSS. Šest vektorů v SSS určuje odsazení pozice vzorků centrálních diferencí oproti osvětlovanému vzorku – $(\pm h, 0,0)$, $(0, \pm h, 0)$ a $(0,0, \pm h)$. Jejich transformací inverzní modelovou maticí mohou být dopočítány pozice vzorků v SSM. Je zvolen jiný přístup. Těchto šest vektorů má vždy pouze jednu složku nenulovou a tato složka je v absolutní hodnotě stejná. Je zvolen vektor (h, h, h). Inverzní transformace je transformace měřítka převrácenou hodnotou oproti původní transformaci, pro vektory transformaci posunutí nemá smysl provádět a jiné transformace nepředpokládáme. Aplikace inverzní transformace se pak rovná vynásobení vektorů $\left(\frac{1}{w \cdot v}, \frac{1}{h \cdot v}, \frac{1}{d \cdot v}\right)$ a (h, h, h) po složkách a výsledkem je vektor $\left(\frac{h}{w \cdot v}, \frac{h}{h \cdot v}, \frac{h}{d \cdot v}\right)$. Protože koeficient h značí délku v SSS, aby jeho velikost odpovídala velikosti strany jednoho voxelu, je třeba ho vynásobit velikostí voxelu v. Předchozí vypočítaný vektor bude nově odpovídat $\vec{d} = \left(\frac{h_{dif}}{w}, \frac{h_{dif}}{h}, \frac{h_{dif}}{d}\right)$, kde h_{dif} odpovídá poměru velikosti strany voxelu: $h_{dif} = h \cdot v$. Pozice šesti vzorků pro výpočet centrálních diferencí v SSM jsou určeny posunutím pozice vzorku v SSM o složky vektoru $\vec{d}$. Na každé souřadnicové ose jsou vypočteny dva vzorky posunutím pozice na dané souřadnicové ose o danou složku $\vec{d}$ a její zápornou hodnotu. Například výpočet pozice pro centrální diference v SSS na ose x je: $P_{+x} = P_x + \overrightarrow{d_x}$ a $P_{-x} = P_x - \overrightarrow{d_x}$, kde P_{+x} a P_{-x} jsou x-ové souřadnice pozice vzorků odsazené v kladném a záporném směru na ose x, P_x je pozice osvětlovaného vzorku na ose x a $\overrightarrow{d_x}$ je x-ová složka vektoru $\vec{d}$. Ostatní souřadnice nových pozic zůstávají stejné jako souřadnice primárního vzorku. Výpočet probíhá obdobně pro ostatní osy. Výpočet je detailně vidět v ukázce 6, která odpovídá implementaci na fragment shaderu.

Využitá BRDF je Blinn-Phongův osvětlovací model s vynecháním zrcadlové složky. Zbývá ambientní složka zadaná konstantně a difúzní složka, která na výpočet vyžaduje znalost

normály povrchu a směrového vektoru ke světlu. V případě trojrozměrných dat je normála reprezentována vypočtenou centrální diferencí. Vektor ke světlu je dopočten transformací pozice ze SSM do SSS. Výpočet osvětlení pro vzorky, které mají opacitu dostatečně nízkou, nemá zásadní vliv na výsledný obraz, ale snižuje výkon. Proto je lokální osvětlení počítáno pouze pro vzorky, jejichž opacita přesahuje mez 0,05.

```
// ... hlavička
// globální intenzity osvětlení: ambientní, difúzní a spekulární
const vec3 phongI = vec3(0.5, 0.5, 0.0);
// koeficient h pro lokální diference
const float hDif = 0.5;
uniform mat3 model3;
// zdrojová data - přejmenována z textureVolume oproti původní ukázce
uniform sampler3D src;

void main(){
    // přepočet velikosti diference podle velikostí stran objemu
    vec3 difMS = hDif / textureSize(src, 0);
    // ... příprava před cyklem
    for( ... ){
        // ... dílčí kroky
        float phongDif = phongI.y; //inicializace difúzní složky
        if(color.a > 0.05){ // limit pro výpočet
            vec3 locDiffMS = vec3(0, 0, 0); // lokální diference
            // výpočet lokálních diferencí upraveným krokem h (difMS)
            locDiffMS.x =
                (texture(src, vec3(posMS.x+difMS.x, posMS.yz)).r
                - texture(src, vec3(posMS.x-difMS.x, posMS.yz)).r)
                / (2 * hDif);
            locDiffMS.y =
                (texture(src, vec3(posMS.x, posMS.y+difMS.y, posMS.z)).r
                - texture(src, vec3(posMS.x, posMS.y-difMS.y, posMS.z)).r)
                / (2 * hDif);
            locDiffMS.z =
                (texture(src, vec3(posMS.xy, posMS.z+difMS.z)).r
                - texture(src, vec3(posMS.xy, posMS.z-difMS.z)).r)
                / (2 * hDif);
            // přepočet lokální diference do SSS
            vec3 locDifWS = model3 * locDiffMS;
            // přepočet souřadnice vzorku do SSS
            posWS = model4 * vec4(posMS, 1.0);
            // výpočet vektoru ke světlu
            vecToLightWS = normalize(lightPos - posWS.xyz);
            // kosinus úhlu mezi vektorem diferencí a vektorem ke světlu
            float cosDif = dot(-normalize(locDifWS), vecToLightWS);
            phongDif = phongI.y*max(0, cosDif);
        }
        // započtení útlumu světla do výsledné barvy
        outColor.rgb = outColor.rgb +
            (1-outColor.a) * corrected.rgb * (phongDif+phongI.x);

    }
}
```

Ukázka 6 – Aplikace lokálního osvětlovacího modelu na fragment shaderu počítaný pro každý vzorek. BRDF pro počítání lokálního osvětlovacího modelu byl zvolen Blinn-Phongův osvětlovací model. Jeho zrcadlová složka je vynechána.

6.2 Osvětlení objemových dat trasováním kužele

Metoda trasování kužele je již delší dobu zavedená metoda s četným využitím – výpočet globálního osvětlení, výpočet ambient occlusion. Namísto trasování kužele způsobem monte carlo (vysláním velkého množství paprsků ve směrech uvnitř kužele) je kužel trasován pomocí pouze jednoho paprsku. Tento paprsek musí dostatečně reprezentovat své okolí tak, aby dokázal nahradit všechny paprsky, které trasovaný kužel reprezentuje. Vhodná reprezentace všech dat je zajištěna předzpracováním. Po dokončení přípravné fáze mohou být data zobrazena.

6.2.1 Přípravná fáze trasování kužele

V přípravné fázi je potřeba připravit vhodně data, aby bylo zajištěno, že trasovaný paprsek bude reprezentovat svoje okolí a tím se nahradí trasování množství paprsků pouze jedním paprskem. Pro toto nahrazení je využito filtrování dat pomocí Gaussova filtru. Je potřeba vytvořit nový objem dat, do kterých se budou nově vypočtené hodnoty ukládat. Filtrování neprobíhá ze zdrojových dat. Požadavkem na filtrování hodnot je definovaná přechodová funkce, která určí optické vlastnosti materiálu. Pro výpočet je důležitá znalost extinkčních koeficientů / opacit.

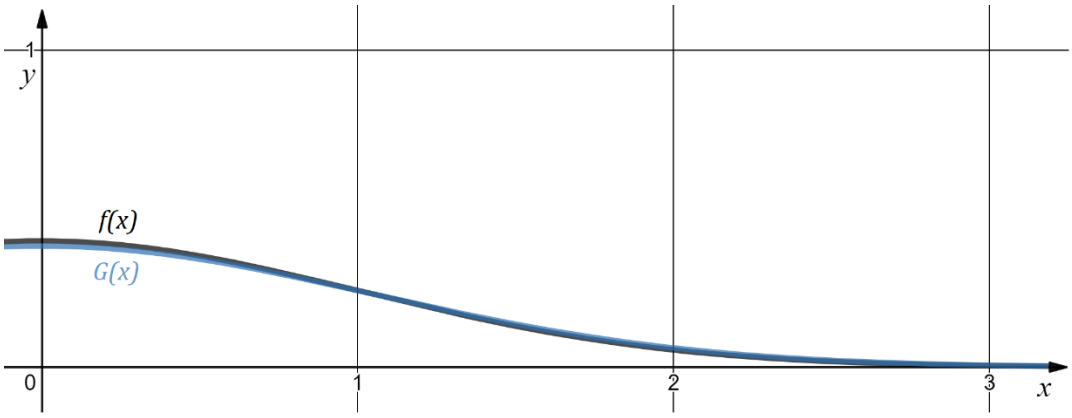
Gaussův filtr je definován následovně pro jeden rozměr:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{x^2}{2\sigma^2}}$$

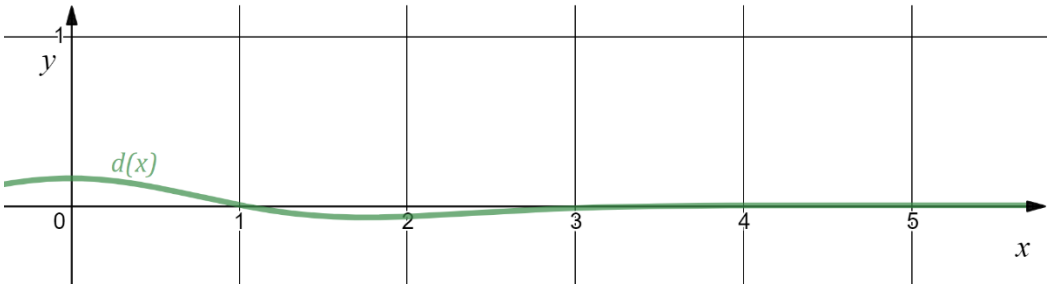
kde σ je směrodatná odchylka. Vlastnosti filtru pro jeden rozměr definují, že v rozpětí od $-\sigma$ do σ je obsaženo 68,27% hodnot, v rozpětí od -2σ do 2σ 95,44% všech hodnot a v rozpětí od -3σ do 3σ 99,73% hodnot. Filtr s větším rozpětím než 3σ ztrácí praktický význam, protože zbylé 0.4% hodnot mají minimální vliv na ostatní hodnoty. Trojrozměrný Gaussův filtr je definován obdobně:

$$f(x) = \frac{1}{(2\pi\sigma^2)^{\frac{3}{2}}} \cdot e^{-\frac{x^2+y^2+z^2}{2\sigma^2}}$$

Aplikací filtru na diskrétní hodnoty je potřeba uvažovat, že hodnoty na celočíselných souřadnicích reprezentují rozsah o velikosti σ. Pro zjednodušení je uvažován případ jednorozměrného filtru. V tom případě hodnota filtru pro $x = 0$ musí reprezentovat interval $\langle\frac{-\sigma}{2}, \frac{\sigma}{2}\rangle$, obdobně pro další hodnoty. Výsledkem distribuční funkce normálního rozdělení na intervalu od $\frac{-\sigma}{2}$ do $\frac{\sigma}{2}$ je 0,3829, to znamená 38,29% hodnot náleží tomuto intervalu. Hodnota normálního rozdělení v bodě $x = 0$ je 0,3989. Tyto hodnoty nejsou stejné, ale jsou dostatečně blízké. Hodnoty distribuční funkce v bodě x s rozsahem 1σ a hodnoty funkce normálního rozdělení v bodě x jsou zobrazeny v grafu na obrázku 18. Jejich rozdíl je zobrazen v grafu na obrázku 19 a hodnoty rozdílů v celočíselných souřadnicích jsou vyneseny do tabulky 1.



Obrázek 18 – Černě f(x) funkce normovaného normálního rozdělení. Modře G(x) distribuční funkce normovaného normálního rozdělení na intervalu $\langle x - \frac{\sigma}{2}, x + \frac{\sigma}{2}\rangle$.



Obrázek 19 – Funkce d(x) zobrazuje 10-ti násobek rozdílu mezi f(x) a G(x) z obrázku 18. $d(x) = 10 \cdot (f(x) - G(x))$

x	0	1	2	3	4	5
$f(x)$	0,3989	0,2420	0,0540	0,0044	0,0001	0,0000
$G(x)$	0,3829	0,2417	0,0606	0,0060	0,0002	0,0000
$d(x)$	0,0160	0,0002	-0,0066	-0,0015	-0,0001	0,0000

Tabulka 1 – Vynesené hodnoty funkcí normálního rozdělní. $f(x)$ je funkce normovaného normálního rozdělení, $G(x)$ je distribuční funkce normovaného normálního rozdělení na intervalu $\langle x - \frac{\sigma}{2}, x + \frac{\sigma}{2}\rangle$, $d(x) = f(x) - G(x)$.

Rozdíl mezi hodnotami těchto funkcí je považován za dostatečně malý. Obdobně to platí pro trojrozměrná data. Díky tomu je filtr počítán z hodnoty funkce normálního rozdělení namísto distribuční funkce.

Filtr je vytvořen jako trojrozměrná textura o rozměrech $(2s + 1)^3$, kde s značí rozsah filtru. Každé odsazení pozice voxelu do strany o jednu jednotku značí odsazení o 1σ na dané souřadnicové ose. Vytvoření filtru probíhá na procesoru počítače. Se známými parametry σ a s lze pro každý voxel vypočítat hodnotu podle vzorce. Výpočet lze rozdělit na dvě části. První část je výpočet mocniny

$$e^{-\frac{x^2+y^2+z^2}{2\sigma^2}}$$

Ta se mění podle měnících se parametrů x, y a z. Konstanta před rovnicí

$$k = \frac{1}{(2\pi\sigma^2)^{\frac{3}{2}}}$$

je společná pro všechny části filtru. Výpočet by pak probíhal jako výpočet mocniny všech voxelů v daném rozsahu a následným vynásobením předpočítaným koeficientem k. Dostatečně velký rozsah filtru je, pokud filtr zahrnuje hodnoty v rozsahu alespoň od -3σ do 3σ. Je žádoucí, aby intenzita hodnoty vrácené filtrem ležela v intervalu $\langle 0,1 \rangle$, kde filtr vrací 0, pokud intenzity veškerých voxelů spadajících pod rozsah filtru jsou nulové. Filtr naopak vrací 1, pokud intenzity všech započítaných voxelů se rovnají 1. Tato podmínka je splněna, pokud je rozsah filtru dostatečně velký. Protože velikost jádra konvoluce filtru je nastavitelná uživatelem, je potřeba zajistit splnění této podmínky i pro menší jádra konvoluce. Koeficient k se rovná inverzní hodnotě součtu hodnot filtru:

$$k = \left(\sum_{x=-s}^{s} \sum_{y=-s}^{s} \sum_{z=-s}^{s} e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \right)^{-1}$$

Implementace filtru je v ukázce 7.

```
// inicializace filtru
double[][][] filter = new double[span*2+1][span*2+1][span*2+1];
// inicializace součtu
double sum = 0.0;

// výpočet části filtru s exponentem
for(int u = -span; u <= span; u++)
    for(int v = -span; v <= span; v++)
        for(int w = -span; w <= span; w++) {
            double value = Math.exp(-(u*u+v*v+w*w)/(2*stdev*stdev));
            filter[u+span][v+span][w+span] = value;
            sum += value;
        }

// normalizace hodnot filtru
for(int u = -span; u <= span; u++)
    for(int v = -span; v <= span; v++)
        for(int w = -span; w <= span; w++)
            filter[u+span][v+span][w+span] /= sum;
```

Ukázka 7 – Implementace vytvoření trojrozměrného Gaussového filtru pro diskrétní mřížku. $stdev$ je označení pro směrodatnou odchylku σ, $span$ je rozsah filtru s, oba parametry jsou zadány uživatelem při generování filtru.

Aplikace filtru na stávající data probíhá pro ušetření výkonu pomocí zobrazovacího řetězce grafické karty. Vykreslování je složeno z přípravné fáze na procesoru a samotné fáze aplikace filtru na zdrojová data, kterým musí být pomocí přechodové funkce přiřazeny vlastnosti.

Standardní výstup vykreslování grafické karty je dvourozměrný buffer s výsledným obrazem. Grafická karta neumožňuje vykreslovat přímo do trojrozměrných dat, výjimkou je

compute shader, který umožňuje zapisovat přímo do textury na celočíselných souřadnicích. Použití vykreslovacího API OpenGL ale umožňuje vykreslovaný buffer přímo nahrávat jako plátek trojrozměrné textury. Vykreslení plátků textury podle osy z umožní sestavení celé trojrozměrné textury. Algoritmus spustí shaderový program pro všechny plátky textury ve směru její hloubky. Aplikace filtru je implementovaná per pixel – shaderový program se spustí pro každý bod plátku. Proto je potřeba zajistit spuštění shaderového programu pro každý pixel výsledného buffer právě jeden krát. Je vykreslován čtverec se souřadnicemi od -1 do 1, jeho viewportovou transformací je roztažen a přesně odpovídá výšce a šířce textury. Jeho rasterizací vzniknou pixely zahrnující všechny voxely daného plátku.

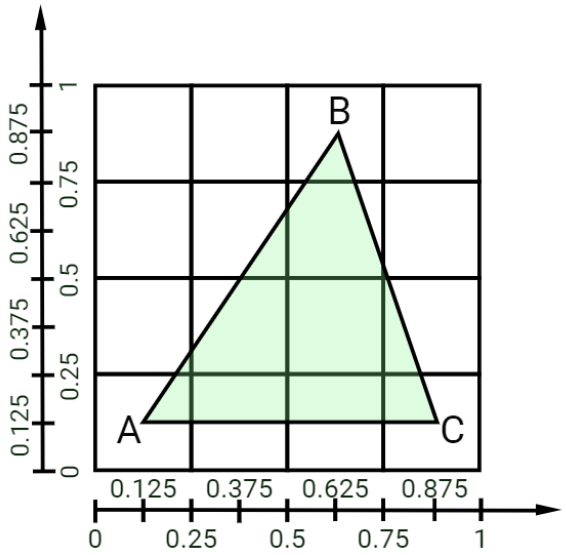
Pro pozdější úsporu při vykreslování jsou rovněž generovány další úrovně mipmapy tohoto objemu. Algoritmus pracuje stejným způsobem pro všechny úrovně, viz ukázka 8. Úrovně mipmap pro primární data jsou generována pomocí algoritmu [31].

```
// inicializace framebufferu - cíl vykreslování
int frameBuffer = glGenFramebuffers();
glBindFramebuffer(GL_DRAW_FRAMEBUFFER, frameBuffer);
// cyklus iteruje přes všechny úrovně mipmap
for(int lod = 0; lod < source.getLODMax(); lod++){
    // nastavení transformace viewport na velikost aktuální mipmapy
    glViewport(0, 0, source.getWidth(lod), source.getHeight(lod));
    // cyklus přes plátky z
    for(int z = 0; z < source.getDepth(lod); z++){
        // nastavení cíle vykreslování jako plátek textury output
        glFramebufferTexture3D(GL_DRAW_FRAMEBUFFER, GL_COLOR_ATTACHMENT0,
            GL_TEXTURE_3D, output.getTextureID(), lod, z);
        // vyčištění případných dat
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        // nastavení relativní souřadnice plátku z na intervalu <0,1>
        glUniform1f(locZ, (float)(z / ((double)source.getDepth(lod)) +
            0.5 / source.getDepth(lod)) );
        // nastavení úrovně mipmapy pro shaderový program
        glUniform1i(locLOD, lod);
        // spuštění vykreslování
        buffer.drawSquare(shaderProgram);
    }
}
```

Ukázka 8 – Příprava vykreslování pro aplikaci filtru na grafické kartě. source je textura vstupních dat, output je textura, do které se zapisuje. locZ a locLOD jsou lokace proměnných shaderového programu.

V ukázce je vidět, že proměnná určující hloubku plátku z není počítána jen dělením této souřadnice hloubkou textury zvětšenou o 1, aby výsledná hodnota ležela v intervalu $\langle 0,1\rangle$. Souřadnice do textury na grafické kartě neuvažují krajní hodnoty 0 a 1. Namísto toho, krajní hodnoty pro jednu souřadnicovou osu jsou $\frac{1}{2d}$ a $\frac{2d-1}{2d}$, kde d je velikost textury na dané souřadnicové ose. Obrázek 20 zobrazuje danou skutečnost v případě rasterizace souřadnic do textury ve 2D. Hodnoty parametru z jsou definovány: $z \in \{0,1,2,\ldots d-2, d-1\}$. Vztah přepočítávající absolutní souřadnici do požadovaného rozpětí je $z_r = \frac{z}{d} + \frac{1}{2d}$, kde z_r je

hodnota parametru z v relativních souřadnicích. Tímto přepočtem $z_r \in \left\{\frac{1}{2d}, \frac{3}{2d}, \frac{5}{2d}, \ldots \frac{2d-3}{2d}, \frac{2d-1}{2d}\right\}$.



Obrázek 20 – Rasterizace souřadnic do textury ve 2D rasterizuje jednotlivé souřadnice tak, že neleží v krajích mřížky. Mřížka reprezentuje mapovanou texturu velikosti 4x4. Souřadnice do textury bodu A ležícího v levém dolním rohu mřížky jsou $(0,0)$ a po rasterizaci jsou přepočítány na souřadnice $\left(\frac{1}{8}, \frac{1}{8}\right)$. Tyto nové souřadnice odpovídají souřadnicím středu texelu levého dolního rohu.

```
ivec3 toAbsolute(vec3 v, sampler3D tex, int lod){
    ivec3 dim = textureSize(tex, lod);
    vec3 off = vec3(1.0)/(2.0*dim);
    vec3 step = vec3(1.0)/dim;
    return ivec3((v-off)/step + 0.5);
}

vec3 toRelative(ivec3 v, sampler3D tex, int lod){
    ivec3 dim = textureSize(tex, lod);
    vec3 off = vec3(1.0)/(2.0*dim);
    vec3 step = vec3(1.0)/dim;
    return off + v*step;
}
```

Ukázka 9 – Funkce přepočítávající absolutní celočíselné souřadnice do relativních a naopak. dim je velikost textury, off je odsazení, $step$ je délka kroku v relativních souřadnicích. Přičítání 0.5 v první funkci je zde kvůli zaokrouhlení. Vstupní parametry: v jsou vstupní souřadnice, tex je textura, pro kterou jsou souřadnice přepočítávány, lod je úroveň mipmapy, pro kterou se souřadnice přepočítávají.

Program na vertex shaderu má vstupní parametry pozici vrcholů, kterou pouze propaguje dále do zobrazovacího řetězce, a souřadnici do textury uv. Z této souřadnice program vytvoří trojrozměrnou souřadnici do textury uvw, kde $w = z_r$ (v programu označovaná jen jako z).

Fragment shader vykonává více činností. V každém spuštění musí tento program načíst hodnoty všech voxelů původní textury, na které zasahuje rozsah filtru. Pro každý tento bod načte jeho intenzitu, přiřadí pomocí známé přechodové funkce optické vlastnosti a na

získanou opacitu bodu aplikuje filtr. Filtr pracuje na celočíselných souřadnicích. Proto je potřeba využít vhodné funkce, která přepočte souřadnice z relativních souřadnic do textury na absolutní ukazatel pozice v textuře a naopak. Tyto funkce jsou implementovány v ukázce 9.

```
#version 330
in vec3 texCoord; // relativní souřadnice voxelu
out vec4 outColor; // výstupní hodnoty

uniform sampler3D source; // zdrojová data
uniform sampler3D gauss; // data filtru
uniform sampler1D tf; // přechodová funkce
uniform int LOD; // úroveň mipmapy

void main() {
    // výpočet rozsahu
    int range = textureSize(gauss, 0).x;
    int span = (range-1)/2;
    // přepočet souřadnic voxelu na absolutní
    ivec3 xyz = toAbsolute(texCoord, source, LOD);
    float value = 0.0; //nasčítávaná hodnota
    for (int u = -span; u <= +span; u++){
        for (int v = -span; v <= +span; v++){
            for (int w = -span; w <= +span; w++) {
                // výpočet relativních souřadnic zdrojové textury
                vec3 sc = toRelative(ivec3(u,v,w) + xyz, source, LOD);
                // výpočet relativních souřadnic textury filtru
                vec3 gc = toRelative(ivec3(u,v,w) + ivec3(span), gauss, 0);
                // intenzita z textury
                float intensity = textureLod(source, sc, LOD).r;
                // aplikace přechodové funkce a filtru
                value += texture(tf, intensity).a * texture(gauss, gc).r;
            }
        }
    }
    outColor = vec4(value, 1.0, 1.0, 1.0);
}
```

Ukázka 10 – Aplikace filtru na fragment shaderu. Podle velikosti filtru jsou spuštěny tři vnořené cykly. Uvnitř cyklů se určují souřadnice voxelů zasažených rozsahem filtru a následně se přepočítají na relativní. V posledním kroku cyklu se aplikuje na získanou intenzitu přechodová funkce a filtr a hodnota se přičte k součtu, který je vrácen jako výstupní hodnota. Ta se uloží do aktuální pozice v objemové textuře.

Jakmile je možný přepočet mezi absolutními a relativními souřadnicemi, je možné provést celý výpočet. Pro každý spuštěný program na fragment shaderu je známá relativní pozice voxelu, pro který se program spustil jako vstup z vertex shaderu. Tato souřadnice je přepočtena na absolutní. Z velikosti textury reprezentující filtr je vypočítán rozsah filtru. Následně je pro každou souřadnicovou osu iterován cyklus v rozpětí $\langle -s, s \rangle$, kde s je rozsah filtru. Tyto cykly jsou vzájemně vnořené. Uvnitř těchto cyklů jsou vypočítány absolutní souřadnice všech okolních voxelů započítaných do filtrování. V dalším kroku jsou souřadnice přepočítány na relativní jak pro zdrojová data, tak i pro filtr. Na zdrojová data je aplikována přechodová funkce a na vrácenou opacitu je aplikován filtr. Výsledná hodnota je naakumulována a přiřazena do výstupní proměnné $outColor$ do její složky r. Program

fragment shaderu je implementován v ukázce 10. Cykly fragment shaderu nejsou omezeny, aby nezasahovaly do souřadnic textur ležící mimo rozsah textur. Proto pro správnou funkčnost je pro zdrojovou texturu nastaven okraj. Jakmile je souřadnice mimo rozsah textury, funkce $texture$ i $textureLod$ navrátí hodnotu nastavenou na okraj. V přípravné fázi jsou okraje nastaveny následovně:

```
glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_BORDER);
glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_BORDER);
glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_BORDER);
glTexParameterfv(GL_TEXTURE_3D, GL_TEXTURE_BORDER_COLOR,
    new float[]{0.0f, 0.0f, 0.0f, 0.0f});
```

Ukázka 11 – Nastavení okraje textury.

6.2.2 Aplikace trasování kužele

V této fázi vykreslování jsou všechna potřebná data připravena k vykreslování. Ve zdrojovém kódu aplikace je nutné všechna data (uniform proměnné, textury) nahrát do grafické karty.

Kód spuštěný na grafické kartě rozšiřuje základní implementaci ray castingu a používá zmíněné optimalizační metody. Jedno z rozšíření je počítání měkkých stínů dopadajících k bodu uvnitř objemu, druhým z nich je počítání ambient occlusion. Oba tyto přístupy využívají trasování kužele.

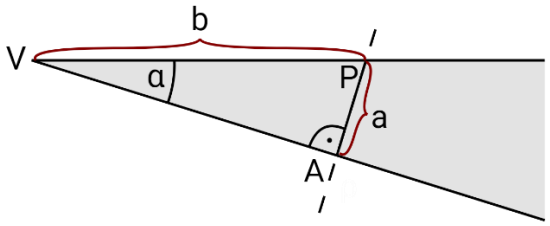
Ray casting prochází celý objem dat pomocí vzorků na diskrétních pozicích vrhaných paprsků. Ne každý vzorek ovlivňuje výsledný obraz. Vzorky s nulovou nebo nízkou opacitou neovlivní výslednou barvu vůbec nebo ji ovlivní pouze v malé míře. Podmínka v ray castingu přeskočí započítání daného vzorku, pokud je opacita menší než uvedená mez – stejný případ jako u lokálního osvětlovacího modelu.

Trasování kužele vyžaduje znalost směru trasování a pozice, ze které má být trasování zahájeno. Pozice v SSM je známá, směr je vypočítán také v SSM. Pro zjištění vržených stínů směrový vektor ke světlu spočítán stejně jako u lokálního osvětlovacího modelu, pro výpočet ambient occlusion je zadaný směr opačný vektor ke směru pohledu dovnitř objemu.

Trasování probíhá ve stylu ray castingu. Probíhá v něm iterační cyklus, kde v každém kroku je načtena hodnota vzorku na paprsku a pomocí kompozitního schématu je akumulována. Optimalizační metoda early ray termination je využita pro zvýšení výkonu.

Využití dalších úrovní mipmap vygenerovaného filtrovaného objemu zvyšuje dále rychlost. Každý voxel další úrovně mipmapy filtrovaného objemu reprezentuje dvojnásobně větší okolí než voxel předchozí úrovně. Nahlížení do dalších úrovní umožní zvýšit délku kroku a současně bude vzorek dostatečně reprezentovat danou část kužele. Na začátku trasování kužele je zvolena nulová úroveň mipmapy a trasování probíhá po zvoleném kroku h. V každém kroku cyklu je podmínka, zda šířka daného okolí je ještě dostatečná, aby reprezentovala danou část kužele. Za dostatečnou je považována šířka odpovídající 2σ. Pokud je tedy v daném kroku vzdálenost od vzorku (paprsek se vzorkem prochází středem kužele) větší než 2σ, nastaví se

vyšší úroveň mipmapy. Protože další úroveň mipmapy reprezentuje dvojnásobně velké okolí, délka kroku h musí být zdvojnásobena. Určení vzdálenosti vzorku ke kraji kužele je spočítáno pomocí úhlu α mezi přímkou procházející středem kužele a přímkou procházející pláštěm kužele. Nechť b je vzdálenost od vrcholu kužele do pozice aktuálního vzorku P, vektor $\vec{v}$ udává směr z vrcholu kužele V středem kuželu do bodu P, bod A je libovolný bod ležící na plášti kužele takový, že přímky $\overleftrightarrow{AP}$ a $\overleftrightarrow{AV}$ svírají pravý úhel, pak a je vzdálenost $|PA|$, viz obrázek 21. Všechny tyto výpočet jsou platné v SSS, který oproti SSM není deformovaný.



Obrázek 21 – Pokud je a větší než 2σ, zvětší se délka kroku stejně jako úroveň mipmapy. V je vrchol trasovaného kužele, P je pozice aktuálního vzorku, A je bod na stěně kužele, mezi nímž a bodem P je počítána vzdálenost.

```
// vstupní parametry jsou pozice a směr v SSM a sin úhlu alfa
float traceConeInDirection(vec3 posMS, vec3 dirMS, float sinus){
    // inicializace proměnných délky kroku podle uživatelem zadaných h a h0
    float locH = h; float locH0 = h0;
    int iInc = 1; // inkrement cyklu
    float lod = 0; // inicializace úrovně mipmapy
    float outIntensity = 0.0; // výstupní hodnota
    float stdevLimit = stDev; // limit pro zvětšení kroku trasování
    int maxIterations = int(length(volDim)/h+1); // limit iterace
    for(int index = 0; index < maxIterations; index += iInc){
        // vypočtení pozice v SSM, dirMS je již vynásobená velikostí voxelu
        vec3 pullMS = posMS + (index * locH + locH0) * dirMS;
        // ... early ray termination
        // načtení zastínení
        float attenuation = textureLod(filteredVol, pullMS, lod).r;
        // aplikace kompozitního schématu
        outIntensity = outIntensity + (1-outIntensity)*attenuation;
        // early ray termination - druhá část
        if(outIntensity > 0.95)
            break;
        // výpočet vzdálenosti a
        float a = sinus * (index * locH + locH0);
        // limit pro zvětšení kroku
        if(a > 2*stdevLimit*voxelSize){
            stdevLimit = 2*stdevLimit; // zvětšení limitu
            lod += 1; // úroveň mipmapy se zvedá
            iInc = iInc*2; // zdvojnásobení délky kroku
        }
    }
    // vrácená hodnota je útlum světla ve směru kužele
    return outIntensity;
}
```

Ukázka 12 – Funkce implementovaná na fragment shaderu pro trasování kužele. h, h_0 a $stDev$ (směrodatná odchylka σ) jsou do programu nahrané jako uniform proměnné.

Podle definice goniometrických funkcí $sin(\alpha) = \frac{a}{b} => a = sin(\alpha) \cdot b$. Úhel je zadán uživatelem a je neměnný pro celý průběh vykreslování. Sinus úhlu alfa je spočítán pro každý snímek vykreslování a nahrán jako uniform proměnná do shaderového programu.

Každý voxel vytváří stín, ale stín vržený tímto voxelem nemá vliv na osvětlení sebe sama. Sám na sebe nevrhá stín. Protože ve filtrovaném objemu každý voxel reprezentuje určité okolí, je potřeba začít trasovat paprsek s odsazením odpovídajícím uživatelem zadané hodnotě h_0. Implementace je zobrazena v ukázce 12.

6.3 Vykreslování pomocí předpočítaných stínů

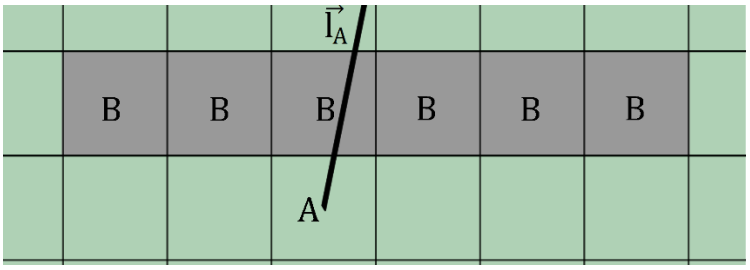
Implementace globálního osvětlovacího modelu popsaná v předchozí kapitole určuje stínování každého bod objemu podle útlumu světla přicházejícího od zdroje světla, jehož utlumení probíhá v kuželu. Při tomto výpočtu ale platí, že mnoho trasovaných kuželů se navzájem překrývá. Proto jako druhá možnost vykreslování je implementováno stínování s využitím předpočítaného stínového objemu. Tato fáze vykreslování opět spočívá ve dvou krocích. První z nich je předpočítání stínového objemu, v druhé fázi probíhá samotné vykreslování.

6.3.1 Výpočet stínového objemu

Nechť $\overrightarrow{l_A}$ je vektor ke světlu L z bodu A ležícího uvnitř objemu, $\overrightarrow{l_B}$ je vektor ke světlu L z bodu B také ležícím uvnitř objemu. Pokud vzdálenost bodu A od světla L je větší, než vzdálenost bodů B a L, a současně bod B leží přesně na dráze nebo dostatečně blízko dráhy paprsku světla z bodu A ve směru $\overrightarrow{l_A}$, potom zastínění bodu A od zdroje světla L je ve větší či menší míře závislé na zastínění bodu B. Diskretizací uvažovaných veličin tak, že body reprezentují voxely, každý voxel objemu má svoji danou velikost, pak zastínění jednoho voxelu od voxelů ležící blíže ke světlu lze zapsat vztahem:

$$U(A) = \sum_{B \in V} u(A, B, \overrightarrow{l_A})$$

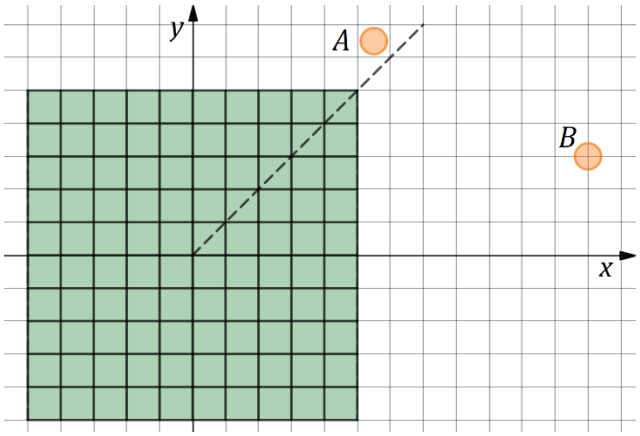
kde $U(A)$ je útlum světla pro počítaný voxel A, B je voxel z dané množiny V, $u(A, B, \overrightarrow{l_A})$ je funkce vracející útlum světla v závislosti na pozici voxelů A a B a vektoru $\overrightarrow{l_A}$. V je množina bodů, které leží blíže ke zdroji světla než bod A. Všechny body množiny V leží v jednom plátku objemu, který je orientován podle vektoru ke světlu $\overrightarrow{l_A}$, a jejich vzdálenost od průniku vektoru s daným plátkem je dostatečně malá, viz obrázek 22.



Obrázek 22 – Body B jsou body/voxely započítávající se do osvětlení bodu A. Všechny leží v plátku objemu orientovaném podle vektoru ke světlu $\overrightarrow{l_A}$ a současně jsou dostatečně blízko průniku vektoru $\overrightarrow{l_A}$ a daného plátku.

Takto vypočtený útlum světla zůstane neměnný, pokud se nezmění pozice světla vůči pozici voxelů. To je v případě, že se nezmění pozice světla a současně se nezmění modelovací transformace objemu (jiné zdroje světla, než bodový zdroj se neuvažují). Situace, že by se současně transformovala pozice světla v SSS i obálka objemu tak, aby relativní pozice vůči sobě navzájem zůstaly nepozměněny, se neuvažuje.

Světlo může být umístěno libovolně v prostoru s výjimkou situace, že by světlo leželo uvnitř obálky tělesa. Pro celý objem je zvolen jednotný směr propagace světla od zdroje do všech voxelů v tělese. Tento směr propagace probíhá ve směru pouze jedné souřadnicové osy. Jinými slovy plátek textury za plátkem v daném směru. Tento směr je určen podle vektoru ke světlu $\vec{l}$ z bodu ležícího uprostřed tělesa. Při transformacích zmíněných v kapitole 6.1.3 (centrování obálky a škálování podle velikosti objemu a velikosti voxelu) se jedná o bod $[0,0,0]$. Složka vektoru $\vec{l}$, která je v absolutní hodnotě největší, určí, podle které souřadnicové osy propagace proběhne. Znaménko této složky určí směr propagace. Například vektor ke světlu $\vec{l} = (-5,4,-1)$ má v absolutní hodnotě největší složku x, takže propagace proběhne podél této osy. Tento člen je současně záporné číslo, tudíž směr propagace bude záporný. Jak se určuje směr propagace je zobrazeno také na obrázku 23.



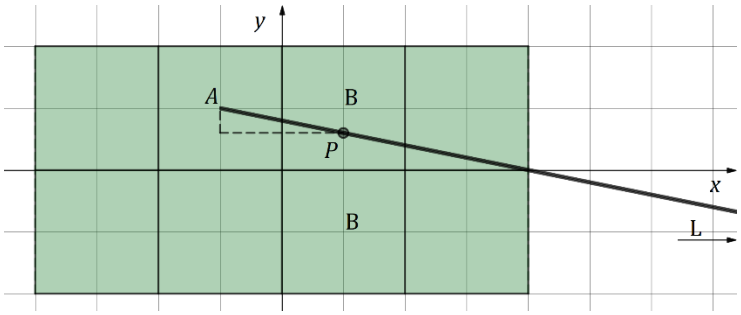
Obrázek 23 – Zelená oblast představuje voxely objemu. Pro měnící se pozici světla se mění také směr propagace světla. Pro světlo v bodě A bude světlo propagováno ve směru $-y$, pro světlo v bodě B zase ve směru $-x$. Pro zjednodušení je situace zobrazena ve 2D. Osy kvadrantů rozdělují celý graf na 4 možné směry propagace. Z obrázku lze vidět, že bod A leží nad osou kvadrantu a světlo je propagováno odlišně oproti světlu v pozici B.

Množina V je díky tomuto směru propagace omezena pouze na část všech voxelů – na voxely plátku ležícího blíže ke světlu. Započítávat všechny voxely předchozího plátku je jak výpočetně náročné, tak bez účinku. Voxely předchozího plátku příliš vzdálené od paprsku $\overleftrightarrow{AL}$ mají zanedbatelný vliv na zastínění voxelu A. Z toho důvodu se množina omezí pouze na voxely v daném vzdálenostním limitu.

Pro výpočet je suma přepsána do jiné formy:

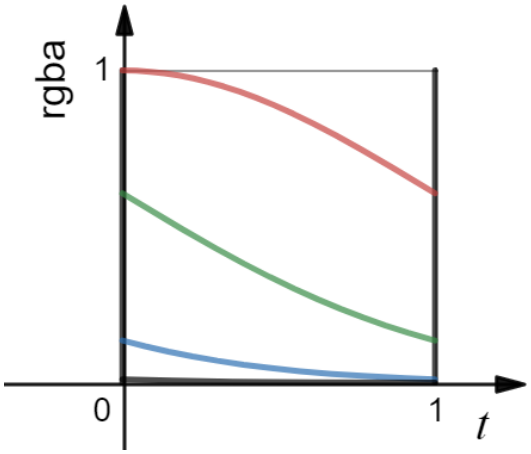
$$U(A) = \sum_{B \in V} g(B, P)$$

kde P je pozice průsečíku paprsku $\overleftrightarrow{AL}$ s rovinou procházející předchozím plátkem (předchozí znamená proti směru propagace), funkce $g(B, P)$ vrací zastínění bodu v závislosti na vzájemné pozici dvou bodů – B a P – s rostoucí vzdáleností klesá hodnota funkce g. Znázornění na obrázku 24.



Obrázek 24 – Výpočet bodu P a označení bodů B. Bod P leží na předchozím plátku propagace světla L ležícího mimo obrázek. Směr je znázorněn šipkou. Pro přehlednost znázornění je obrázek ve 2D.

Funkce g je implementována jako Gaussův filtr. Namísto počítání hodnot filtru v celočíselných souřadnicích je v tomto případě funkce závislá na vzdálenosti mezi body B a P. Pro souřadnicovou osu propagace světla je vzdálenost mezi bodem P a body B nulová.



Obrázek 25 – Uložení hodnot Gaussova filtru do barevných komponent textury. t je souřadnice do textury, intervaly $\langle k\sigma, (k+1)\sigma\rangle$ jsou uloženy do barevných komponent, $k \in \{0,1,2,3\}$.

Bod A je spočítán jako suma intenzit všech bodů B vynásobená hodnotou filtru pro daný bod. Počítání hodnoty filtru není rychlá operace. Proto není vhodné počítat pro každý bod B

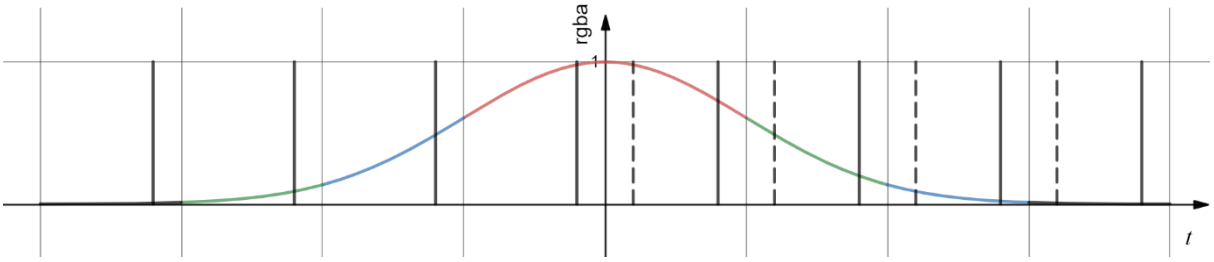
jeho hodnotu. Namísto toho je zvoleno uložení hodnot filtru do jednorozměrné textury. S dostatečnou šířkou této textury budou uložené hodnoty dostatečně přesné a chybějící hodnoty mohou být dopočteny lineární interpolací. Pro vytváření textury filtru je využita sudost funkce normovaného normálního rozdělení. Hodnoty funkce ležící v záporné části osy x mohou být snadno dopočítány z kladné části osy. Dále je využito ukládání dat do jednotlivých komponentů textury. Textura (v OpenGL) má maximum až 4 barevné komponenty – rgba. Pro každou komponentu je uložena hodnota Gaussova rozdělení v rozsahu $\langle k\sigma, (k+1)\sigma\rangle$, kde k značí index barevné komponenty podle pořadí, $k \in \{0,1,2,3\}$, viz obrázek 25.

Jak bylo zmíněno v kapitole 6.2.1, ukládat data do objemové textury není umožněno přímo. V předchozím řešení bylo využito vykreslování do dvojrozměrného bufferu, který se aplikoval jako plátek trojrozměrné textury. Tato metoda umožňuje vykreslovat pouze do plátků v rovině z. Protože propagace světla může probíhat po všech třech souřadnicových osách na každé z nich v kladné i záporném směru, je zvolena metoda generování pomocí compute shaderů v OpenGL. Tato shaderová jednotka je nezávislá na standardním vykreslovacím řetězci. V programu spuštěném compute shaderem nejsou žádné vstupní a ani výstupní parametry, pouze data – buffery a obrázky (Rozdíl oproti textuře je, že z obrázků a do obrázků lze v compute shaderu přistupovat na celočíselných souřadnicích. Textura je určena ke čtení dat podle souřadnic do textury v rozsahu $\langle 0,1\rangle$.), ze kterých shader čte data a do kterých data zase zapisuje. Compute shader umožňuje číst data z libovolných pozic a zase zapisovat do libovolných pozic.

Aby byla zajištěna postupná propagace světla v daném směru, je potřeba plátek po plátku počítat zastínění následujícího plátku. Transformace obálky tělesa jsou neměnné, střed tělesa leží v bodě $[0,0,0]$. Podle maximální hodnoty vektoru ke světlu a znaménka této hodnoty je určen vektor $\vec{d}$ jako jeden z šesti možných – $(\pm 1,0,0)$, $(0,\pm 1,0)$ nebo $(0,0,\pm 1)$. Pro spuštění výpočtů pro každý plátek na compute shaderu je spuštěn cyklus, který iteruje právě po dané ose v daném směru propagace světla. Compute shader je pak spuštěn v každé iteraci cyklu pro všechny voxely daného plátku. Aktuální iterátor cyklu je nahrán jako uniform proměnná do výpočtu.

Výpočet na compute shaderu probíhá následovně: Pozice aktuálního voxelu je známá, stejně tak pozice světla, směr propagace světla, modelovací a inverzní modelovací transformace, zdrojová data, hodnoty vypočtené v předchozích plátcích, textura pro přechodovou funkci i textura pro filtr. Aktuálně počítaný voxel se transformuje první do relativních souřadnic v SSM (interval $\langle 0,1\rangle$), následně se transformuje modelovou transformací do SSS. Z této souřadnice je vypočten směrový vektor ke světlu. Následuje výpočet pozice P jako průsečíku předchozího plátku s paprskem ke zdroji světla. Aby algoritmus fungoval obecně pro všechny případy možné propagace světla, je parametr t definován jako trojrozměrný vektor. Každá složka z vektoru je získána jako desetinná část dané složky souřadnice bodu P. $t = 0$ pro souřadnicovou osu, po které probíhá propagace světla.

V je množina bodů B. Souřadnice bodů B na souřadnicové ose propagace je stejná pro všechny body. Pro ostatní souřadnicové osy je do výpočtu zahrnuto 8 nejbližších hodnot na každé souřadnicové ose, dohromady tedy 8x8 hodnot. Pro jednu souřadnicovou osu leží souřadnice bodů na intervalu $\langle floor(P_0) - 3, floor(P_0) + 4\rangle$, kde P_0 označuje hodnotu počítané souřadnice daného bodu, $floor$ je označení pro zaokrouhlení dolů (funkce vrací pouze celočíselnou část čísla). 8 hodnot pro každou souřadnicovou osu odpovídá 2x4 hodnotám, které lze získat z filtru uloženého do textury. Díky zrcadlení funkce budou hodnoty v záporné části grafu rovny hodnotám v kladné části grafu. Na body B s danou souřadnicí menší, než daná souřadnice bodu P, bude aplikován filtr v hodnotě $1 - t$, na druhé 4 hodnoty bude aplikován filtr v souřadnici t, viz obrázek 26. Celý algoritmus implementovaný v compute shaderu je v ukázce 13. Kvůli obsáhlosti implementace je ukázka vložena na konec této práce.



Obrázek 26 – Graf zobrazuje souřadnici to textury t ležící v intervalu $\langle 0,1\rangle$. Tmavé svislé úsečky značí hodnoty vzorků pro jednotlivé barevné komponenty. V záporné části osy t lze pro hodnoty t (plná svislá úsečka) určit návratovou hodnotu souřadnicí $1 - t$ (přerušovaná svislá úsečka). Každý barevný komponent leží na intervalu $\langle 0,1\rangle$.

6.3.2Vykreslování s předpočítaným stínem

Vykreslování pomocí předpočítaného stínového objemu je velmi obdobné předchozí metodě pomocí trasování kužele. Základ je opět stejný, jedná se i implementaci ray castingu s využitím zmíněných optimalizačních metod. Pro každý vzorek raycastingu, pokud je jeho opacita větší než daná mez, aplikuje se osvětlení pomocí stínového objemu.

Aby nedocházelo k samozastínění voxelů, je potřeba aplikovat osvětlení z pozice odsazené o h_0 ve směru vektoru ke světlu. Výpočet vektoru ke světlu je nezměněn. Pokud je délka vektoru ke světlu v SSS rovna zadané velikosti voxelu, pak po transformaci vektoru do SSM je vektor vynásoben délkou kroku h_0, který udává délku kroku v počtu voxelů.

7 Výsledky

V předchozích kapitolách jsou popsány implementace dvou metod globálního osvětlovacího modelu aplikovaného na objemová data. První metoda předpočítává data tak, aby vhodně reprezentovala své okolí a trasováním kužele pak z těchto dat počítá vržené stíny a ambient occlusion. Druhá – autorská metoda předpočítává celý objem vržených stínů do nové objemové textury plátek po plátku. Obě tyto metody jsou porovnávány dle vybraných kritérií. Protože cíl této práce je primárně aplikace osvětlení v reálném čase, hlavním z kritérií je rychlost vykreslování. Samotné přípravné fáze předzpracování objemů jsou náročné. Proto je diskutována i tato výpočetní cena, která se ale při dalších překreslení neprojeví. Většina renderovacích metod řeší kompromis mezi rychlostí překreslení a kvalitou zobrazení. Proto je další kritérium vizuální kvalita výsledného obrazu. Dalším kritériem jsou paměťové nároky jednotlivých algoritmů a v případě objemových dat to není ani zdaleka zanedbatelná položka. Velikost uložených dat roste s třetí mocninou velikosti strany (Při neměnném poměru délek stran.). Proto je diskusi na téma paměťových nároků také věnována část výsledků. Pro veškeré testování byl využit notebook s procesorem Intel Core i7 4710HQ Haswell a grafickou kartou NVIDIA GeForce GTX 860M. Zvolené rozlišení pro vykreslování je 1200x900 a je konstantní pro všechny testy.

Aplikace klade také požadavek na zvolení libovolné přechodové funkce uživatelem. Nástroj na její definici je součástí aplikace.

7.1 Výběr přechodové funkce

Zvolení přechodové funkce (viz kapitola 4) je zcela ponecháno na uživateli. Středem zájmu může být pro každého uživatele něco jiného a stejně tak to může být odlišné pro každý datový model. Proto se přechodová funkce může měnit zcela libovolně. Pro aplikaci se funkce uloží do jednorozměrné textury.

Zadávání uživatelem probíhá interaktivně a uživatele nelimituje. Uživatel má k dispozici nástroj, kterým přechodovou funkci graficky kreslí. Nekreslí ji ale po jednotlivých texelech. Namísto toho má k dispozici pomocné body, které může v libovolném množství vytvářet a mazat. Nástroj na výběr přechodové funkce se skládá z tří modulů.

První modul nástroje umožňuje uživateli volit aktuálně upravovanou barvu (včetně opacity), kterou bude používat na úpravu přechodové funkce v druhém modulu. Výběr barvy ovlivňuje, které pomocné body budou upravovány, vytvářeny či mazány. Krom výběru barev je k dispozici možnost nahrání a uložení přechodové funkce z a do souboru. Následuje možnost změnit rozsah – uživatel má možnost ovlivnit šířku textury, do které se přechodová funkce uloží. Poslední část modulu nahraje aktuálně vytvářenou přechodovou funkci do textury.

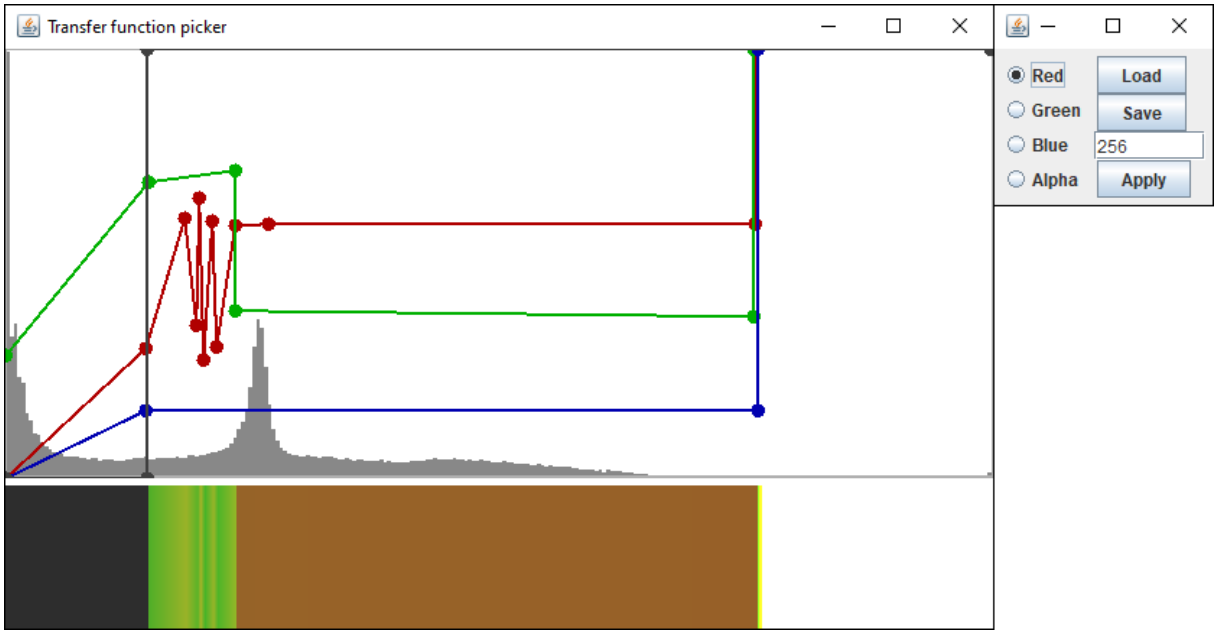
Druhý modul umožňuje samotnou volbu přechodové funkce. Uložení přechodové funkce probíhá v relativních souřadnicích uložených ve formátu plovoucí desetinné čárky.

Každý bod má dvě souřadnice, z nichž x značí pozici a y je hodnota dané barevné komponenty přechodové funkce na dané pozici. Obě souřadnice xy leží v intervalu $\langle 0{,}1\rangle$. Každá barevná komponenta je uložena zvlášť, pro každou z nich platí, že modul výběru obsahuje vždy minimálně dva body, které jsou krajními a mají hodnoty $x = 0$ a $x = 1$. Libovolné množství bodů mezi těmito dvěma body může být vytvořeno, následně editováno (změnou pozice x i y) a případně mazáno. Dvěma krajním bodům může být editována pouze hodnota y. Zobrazení modulu pro uživatele spočívá z transformaci souřadnic bodů z intervalu $\langle 0{,}1\rangle$ na interval $\langle 0, w\rangle$ nebo $\langle 0, h\rangle$ – w je šířka plátna, h je výška plátna, na které se body vykreslují a s kterým uživatel interaguje. Protože počet bodů neodpovídá počtu texelů výsledné textury, veškeré chybějící hodnoty jsou lineárně interpolovány z hodnot nejbližších sousedních bodů.

Součástí tohoto modulu je také zobrazení histogramu. Při načtení nových dat se přepočítá a zobrazí jako pozadí toho modulu pro lepší uživatelskou představu o tvorbě přechodové funkce.

Třetí modul slouží jako náhled aktuálně upravované přechodové funkce. Oblast je rozdělena na w úseček se zadanou výškou h_d, kde w je absolutní šířka modulu v pixelech. Pro každou úsečku je lineární interpolací spočítána barva podle uživatelem zvolené přechodové funkce z druhého modulu. Protože přechodová funkce umožňuje měnit také opacitu, barva pozadí tohoto modulu osciluje mezi černou a bílou, díky čemuž vyniknou průhledné části přechodové funkce.

Modul 2 a 3 tvoří jeden celek. Všechny tyto tři moduly jsou zobrazena na obrázku 27.



Obrázek 27 – Vpravo: první modul výběru přechodové funkce. Umožňuje zvolit upravovanou barvu, načítat a ukládat přechodové funkce, měnit velikost textury a do které se uloží. Vlevo horní část: druhý modul umožňuje uživateli upravovat přechodovou funkci pomocí nastavitelných volitelných bodů (šedé body jsou pro opacitu). Jako pozadí je histogram aktuálních dat. Vlevo dolní část: třetí modul vytváří náhled přechodové funkce. V levé části modulu je tmavá barva, která prosvítá z měnícího se pozadí díky nulové opacitě v dané části. Aktuálně zobrazená funkce je využitá při zobrazení datasetu Bonsai, viz následující podkapitola.

7.2 Parametrizace a datasety

Parametry pro vykreslované metody značně ovlivňují jak výsledný obraz, tak rychlost vykreslování. V této podkapitole je věnována pozornost jednotlivým nastavitelným parametrům pro implementované metody. Některé tyto parametry jsou nastavitelné pouze pro jednu metodu globálního osvětlení nebo optimalizační metodu. Pokud není později řečeno jinak, parametry použité v testech jsou ty uvedené zde.

- f_v značí frekvenci vzorkování primárního paprsku v závislosti na velikosti strany voxelu. S využitím metody adaptive samplingu $f_v = 1{,}5$.
- f_{as} je násobek vzorkovací frekvence při adaptive samplingu. Jinými slovy tento parametr říká, kolik vzorků mezi dvěma následujícími vzorky bude započítáno. S využitím adaptive samplingu $f_{as} = 5$.
- σ je směrodatná odchylka použitá pro výpočet filtrovaných hodnot. Pro obě metody je tato hodnota využita při generování filtru a je nastavena na hodnotu 1.0.
- s značí rozsah Gaussova filtru pro metodu trasování kužele. Pro metodu předpočítaných stínů tento parametr nastavitelný není, kvůli uložení textury do jejích 4 barevných komponent. Pro trasování kužele je rozsah nastaven na 3σ, který zahrnuje většinu hodnot.
- h_0 určuje vzdálenost prvního kroku při trasování kužele. Pro druhou metodu tento parametr udává odsazení vzorku, ze které se bude načítat hodnota zastínění. Tento parametr zabraňuje zastínění vzorku sebou sama. Pro trasování kužele je nastavena na vzdálenost 3σ, za kterou v Gaussově rozdělení leží minimum hodnot, a pro předpočítané stíny je nastavena na 1.25.
- h je délka kroku pro trasování kužele v nulté úrovni mipmapy. Délka kroku se dále může zvětšovat. Je nastavena na hodnotu 2.5σ.
- α úhel kužele při trasování směrem ke světlu. Nastaven na 5°.
- β úhel kužele při trasování směrového ambient occlusion. Nastaven na 30°.

Výsledky použitých metod byly testovány na třech datasetech lišících se velikostí i obsahem dat. První z nich je dataset Bonsai s velikostí 256^3, druhým je Lobster o velikosti 301x324x56 a poslední testovaný dataset je Backpack s velikostí 512x512x373. Zdroj datasetů je uveden v kapitole 9.

7.3 Výsledky vykreslování

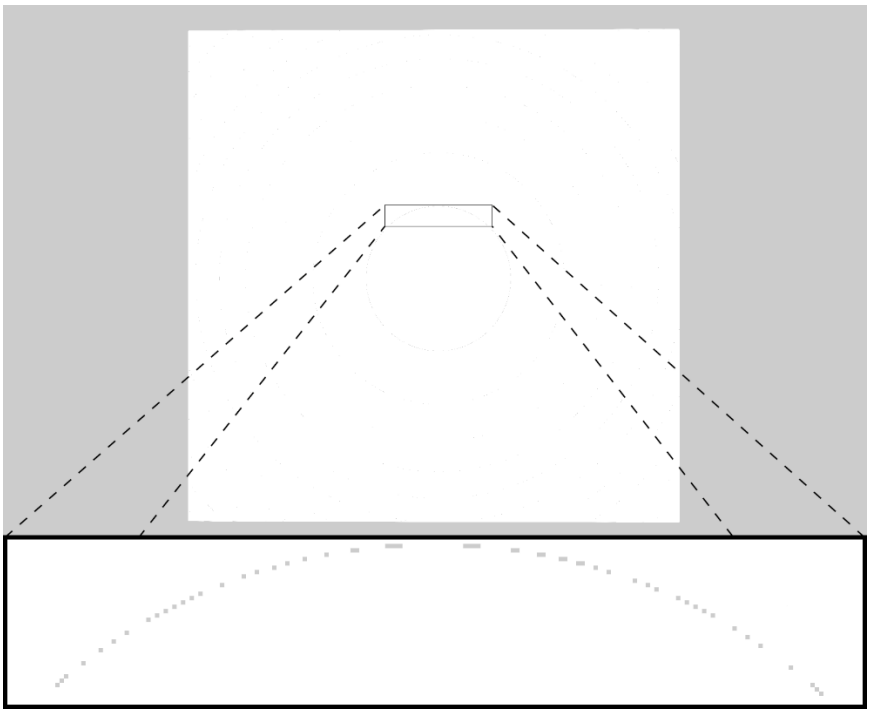
Mimo dvou metod globálního osvětlovacího modelu implementovaných v této práci byl implementován také lokální osvětlovací model a optimalizační metody. Ty mají za cíl zvýšit rychlost vykreslování nebo kvalitu výsledného obrazu.

První z testovacích metod je metoda early ray termination, která je dnes již standardem. Její testování bezprostředně dokazuje její účinnost. Testovány byly 4 různé variace. První z nich bylo vykreslování bez této metody, druhou možností bylo testování pouze s podmínkou, která ukončuje trasování paprsku, pokud je libovolná souřadnice vzorku mimo interval $\langle 0{,}1\rangle$. Třetí

možnost bylo vykreslování s druhou podmínkou, která ukončuje trasování, pokud naakumulovaná opacita překročí danou mez. Poslední možností bylo vykreslování s oběma podmínkami. Přechodová funkce byla nastavená na plně neprůhlednou od dané meze tak, aby data vystihovala svůj obsah. Testování ukazuje, že využití této metody zvyšuje počet vykreslovaných snímků za vteřinu (FPS). V závislosti na testovaném datasetu zvýšila druhá možnost rychlost vykreslování přibližně 2x až 3x. Třetí možnost neukázala takový nárust, nicméně stále zvýšila rychlost průměrně o 27%. Použití obou částí metody zvýšilo FPS 3x až 4x. Podrobné výsledky jsou zapsány v tabulce 2. Tyto výsledky ukazují, že využití této metody, která je z implementačního hlediska značně jednoduchá, je efektivní způsob zvýšení FPS a její použití v této práci má významný vliv na výslednou rychlost. Testování dalších metod už vždy předpokládá použití této metody.

Dataset	**Bonsai**		**Lobster**		**Backpack**	
Podmínka ukončení	FPS	nárůst	FPS	nárůst	FPS	nárůst
Žádná	8,65		15,71		2,21	
Paprsek opustí obálku	19,15	2,21x	51,40	3,27x	4,29	1,94x
Opacita překročí mez	12,80	1,47x	17,43	1,10x	2,77	1,25x
Obě metody současně	28,28	3,26x	62,84	3,99x	6,59	2,98x

Tabulka 2 – Využití metody early ray termination ukazuje nárust FPS (snímků za vteřinu)



Obrázek 28 – Nahoře: pohled na vykreslený kvádr velikosti 256x256x1 s využitím adaptive samplingu. Velikost vykresleného kvádru je přibližně 900x900 pixelů. Dole: výsledný obraz byl ořezán a zvětšen. Některé pixely výsledného obrazu mají namísto korektní bílé barvy jinou barvu.

Jak dokazují výsledky, využití metody adaptive samplingu dále zvyšuje výkon. U této metody je důležité pozorovat jak zvýšení výkonu, tak kvalitu výsledného obrazu, na který může mít změna vzorkovací frekvence významný vliv. První z testů byl prováděn na testovacích datech, jejichž struktura odpovídá příkladu z obrázku 17. V prázdném datasetu velikost 256^3 byl doprostřed umístěn kvádr velikost 256x256x1. Při pohledu ve směru jeho nejkratší osy byly pozorovány artefakty, kdy část pixelů celkového obrazu je špatně zobrazena, viz obrázek 28. Jejich barva neodpovídá přechodové funkci, která v daném případě vrací pouze bílou barvu. Po zvětšení kvádru na velikost 256x256x2 už se artefakt v obraze nevyskytuje.

Dataset s kvádrem byl vytvořen pro demonstraci a ověření popsaných artefaktů. I když jsou chyby zřetelné, celkové množství pixelů s chybně vypočtenou hodnotou není příliš velké. Další testovací datasety jsou reálně snímané scény, jejichž tělesa nemají dokonale rovný povrch s intenzitou stejnou pro veškeré voxely. Pro tyto datasety nebyly artefakty pozorovány.

Využití adaptivního samplingu představuje znatelný nárůst rychlosti vykreslování. Zvýšení vzorkovací frekvence umožňuje trasovat prázdné a homogenní oblasti menším počtem vzorků. Cenou za tento přístup je jedna operace čtení ze zdrojových dat navíc. V každém kroku je nutné znát intenzitu následujícího vzorku a podle toho určit, zda je potřeba vzorkovací frekvenci zvýšit. Tato operace čtení vzorku navíc je vykompenzována možností přeskakovat větší úseky paprsku. Zvláště pro datasety s velkými prázdnými oblastmi představuje metoda výrazné zlepšení. Pro Bonsai byl naměřen nárůst FPS o 52%, pro dataset Lobster o 21%. a pro Backpack o 143%. Podrobné údaje jsou vyneseny do tabulky 3. Pro testování byla použita vzorkovací frekvence 1,5, která se v případě potřeby zpětinásobí. Výsledky ukazují zlepšení rychlosti vykreslování díky využití této metody. Metoda adaptive samplingu je pak dále uvažována při veškerém dalším vykreslování.

Dataset	**Bonsai**		**Lobster**		**Backpack**	
Vzorkovací frekvence	FPS	nárůst	FPS	nárůst	FPS	nárůst
0,4775x1	28,28		62,84		6,59	
1,5x5	49,41	1,74x	76,09	1,21x	16,06	2,43x

Tabulka 3 – Výsledky využití metody adaptive samplingu. Vzorkovací frekvence v daném formátu AxB znamená: A je primární vzorkovací frekvence (vztaženo na délku strany voxelu), B je násobek primární vzorkovací frekvence pro heterogenní oblasti. V prvním případě je využita neměnné frekvence 0,4775, v druhém případě je vzorkovací frekvence 1,5, která se v případě potřeby zpětinásobí.

Počítání lokálního osvětlení je společné pro obě implementované metody globálního osvětlení. Pro každý vzorek na dráze paprsku od pozice pozorovatele dovnitř objemu, pokud je jeho intenzita větší než daná mez, je počítáno lokální osvětlení s využitím centrálních diferencí. K výpočtu je potřebná intenzita šesti nových vzorků a několik aritmetických operací. Tyto výpočty zvyšují kvalitu výsledného obrazu. Díky nim je lépe zřetelný povrch zobrazovaných dat, stejně tak je zvýrazněn tvar tělesa. Náročnost tohoto výpočtu je závislá ve velké míře na zvolené přechodové funkci a zvolené mezi, od které se lokální osvětlení bude

počítat. V testovacím prostředí tento osvětlovací model snížil rychlost vykreslování o 5-25% podle zvolených dat, průměrovaná data jsou v tabulce 4. Je důležité poznamenat, že přechodová funkce byla vhodně zvolena (vrací opacitu buď 0 nebo 1). Po průniku paprsku do neprázdného prostoru objemu je akumulovaná opacita 1 nebo blízká jedné a metoda early ray termination zabrání dalším výpočtům.

Dataset	**Bonsai**		**Lobster**		**Backpack**	
Lokální osvětlení	FPS	změna	FPS	změna	FPS	změna
Vypnuto	49,41		76,09		16,06	
Difůzní složka	43,60	0,88x	70,22	0,92x	15,28	0,95x

Tabulka 4 – Lokální osvětlovací model snižuje rychlost vykreslování průměrně o 10%. Jeho pomocí je dosaženo lepšího vizuálního vjemu.

7.3.1 Výsledky globálního osvětlení

Metody porovnávané výše jsou společné pro oba modely globálního osvětlení. Pro každý z těchto modelů jsou z hlediska rychlosti vykreslování důležité dvě věci: první z nich je rychlost přípravné fáze a druhou z nich je rychlost vykreslování s předpřipravenými daty.

U obou modelů přípravná fáze spočívá v předpočítání nového objemu dat. Tato příprava se liší způsobem jejich generování a tím, kdy je potřeba ji přepočítat. Pro trasování kužele, pomocný objem je generován pomocí renderování do plátku 3D textury. V metodě předpočítaných stínů je objem generován počítáním na compute shaderu. Krom využité technologie jsou důležité rozdíly v tom, kdy je potřeba přepočítat tento objem. Pro metodu trasování kužele je to potřeba vždy, když se změní zdrojová data nebo pokud se změní přechodová funkce. Druhá metoda vyžaduje přepočítání nového objemu ve více případech. Změna zdrojových dat a přechodové funkce jsou první dva případy, dalšími jsou, pokud se změní vzájemná pozice voxelů a pozice světla. Aby nedocházelo k příliš častému přepočítávání nového objemu, uživatel interaktivně mění pozici pozorovatele namísto rotace obálky tělesa. Časy na přepočítání nejsou zanedbatelné. Jsou vyneseny v tabulce 5.

Dataset	**Bonsai**	**Lobster**	**Backpack**
Počítaný objem	Čas	Čas	Čas
Filtrovaná opacita	0,89s	0,30s	4,99s
Předpočítané stíny	3,59s	0,71s	34,59s

Tabulka 5 – Průměrný čas generování předpočítaných objemů pro metody globálního osvětlení.

Pokud budou přechodová funkce nebo zdrojová data měněny příliš často, aplikace nebude mít plynulý průběh. Je zřejmé, že výsledné časy rostou s třetí mocninou velikosti strany objemu. Pro objem dvojnásobné velikosti na jedné straně lze předpokládat 8-násobnou dobu výpočtu (při neměnném poměru délek stran). Výsledky ukazují, že rychlost generování

filtrované opacity je několikanásobně vyšší. Možnou příčinnou je využití jiného způsobu zápisu do trojrozměrné textury. Pro větší datasety, například pro využitý dataset Backpack, je předpočítání stínů téměř neúnosné.

Při ukládání veškerých textur v této práci na grafickou kartu neprobíhá žádná komprese. Výsledná velikost dat je závislá na rozměrech textury a způsobu uložení dat. Primární data stejně jako filtrovaná opacita pro metodu trasování kužele, jsou ukládány ve formátu 8-bitového nezáporného celého čísla. Předpočítané stíny jsou ukládány s vyšší přesností, aby nedocházelo k chybám z nedostatečné přesnosti uložených dat. Jsou uloženy ve standardním 32-bitovém formátu plovoucí desetinné čárky (*float* podle standardu IEEE 754). Ostatní textury se při porovnání paměťové náročnosti neuvažují. Jejich velikosti je pro větší datasety oproti trojrozměrným texturám zanedbatelná.

Velikosti textur na předzpracovaná data odpovídá velikosti primárních dat. Pro metodu trasování kužele jsou na jeden voxel potřeba 2B místa, pro druhou metodu 5B. Paměťová náročnost pro jednotlivé datasety je uvedena v tabulce 6. Paměťová náročnost pro tyto datasety je únosná, veškerá data byla uložena do paměti grafické karty. Nicméně pro větší datasety by bylo třeba využít kompresních formátů, vhodné úpravy algoritmu předzpracování dat nebo použití textur na tato data s menším rozlišením.

Dataset	**Bonsai**		**Lobster**		**Backpack**	
Paměťová náročnost	Velikost	Součet	Velikost	Součet	Velikost	Součet
Primární data	16,00		5,21		93,25	
Filtrovaná opacita	16,00	32,00	5,21	10,42	93,25	186,50
Předpočítané stíny	64,00	80,00	20,83	26,04	373,00	466,25

Tabulka 6 – Porovnání paměťové náročnosti jednotlivých datasetů a textur s předzpracovanými daty. Ve sloupci sloupec je paměťová náročnost pro daný dataset s využitím dané metody globálního osvětlení. Všechny hodnoty v tabulce jsou uvedeny v MB.

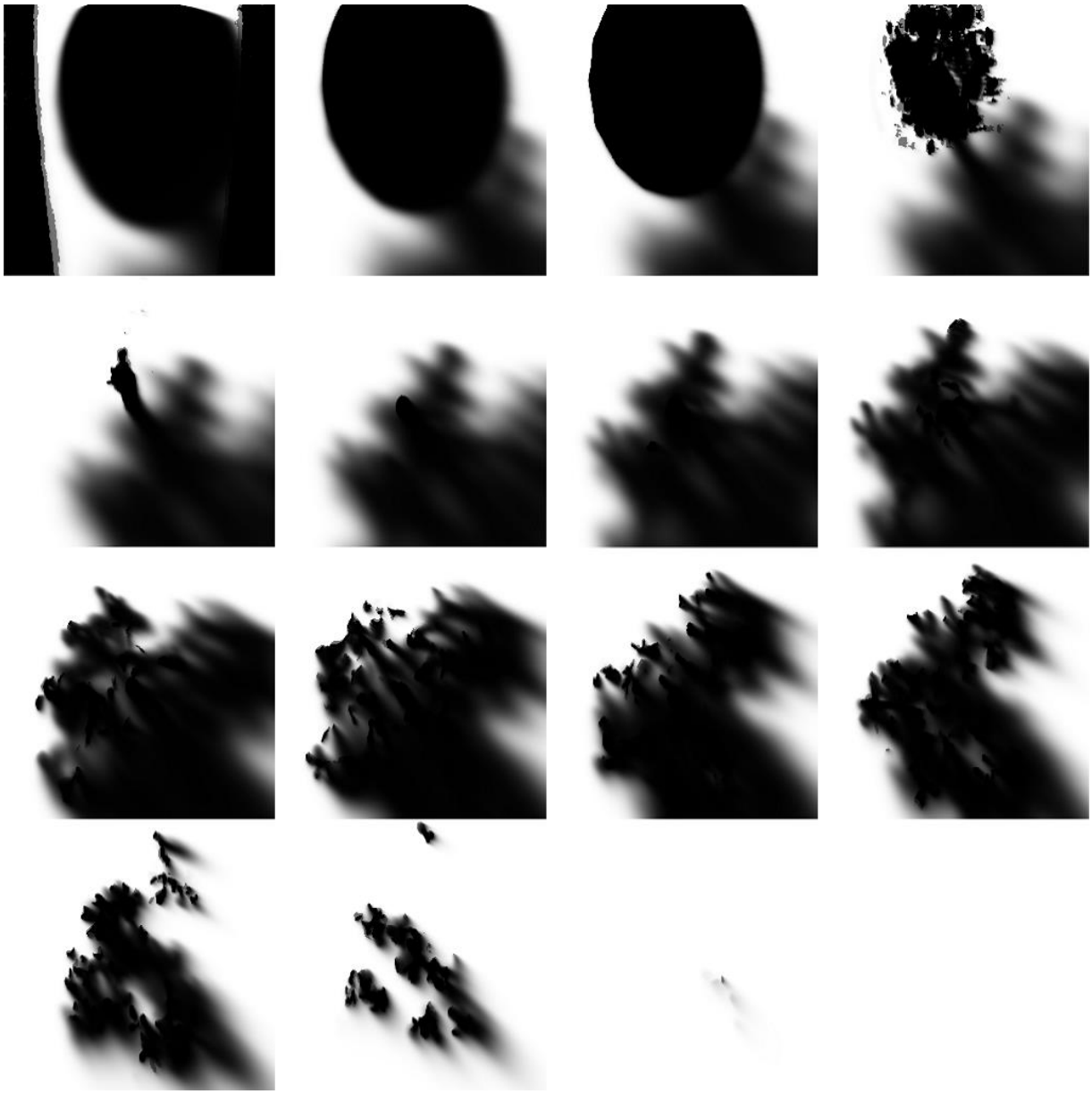
Výsledky samotného vykreslování ukazují, že metoda předpočítaných stínů dosahuje vyšší rychlosti vykreslování. Obrázek 29 ukazuje vypočtený objem stínů. I když metoda trasování kužele značně zjednodušuje celé trasování nahrazením kužele jedním paprskem, její výpočetní náročnost má kvadratickou závislost – pro každý vzorek trasovaného paprsku může nastat sekundární trasování ke zdroji světla. Výpočetní náročnost předpočítaných stínů je lineárně závislá na počtu primárních paprsků – pro stínované vzorky paprsku je načtena pouze jedna hodnota. Výsledky rychlosti vykreslování jsou vyneseny do tabulky 7.

Dataset	**Bonsai**		**Lobster**		**Backpack**	
Osvětlovací model	FPS	rozdíl	FPS	rozdíl	FPS	rozdíl
Pouze lokální	43,60		70,22		15,28	
CT stíny	15,01	0,34x	54,25	0,77x	9,72	0,63x
CT stíny + AO	14,03	0,32x	47,08	0,67x	8,07	0,52x
Předpočítané stíny	41,71	0,95x	69,32	0,98x	14,45	0,94x

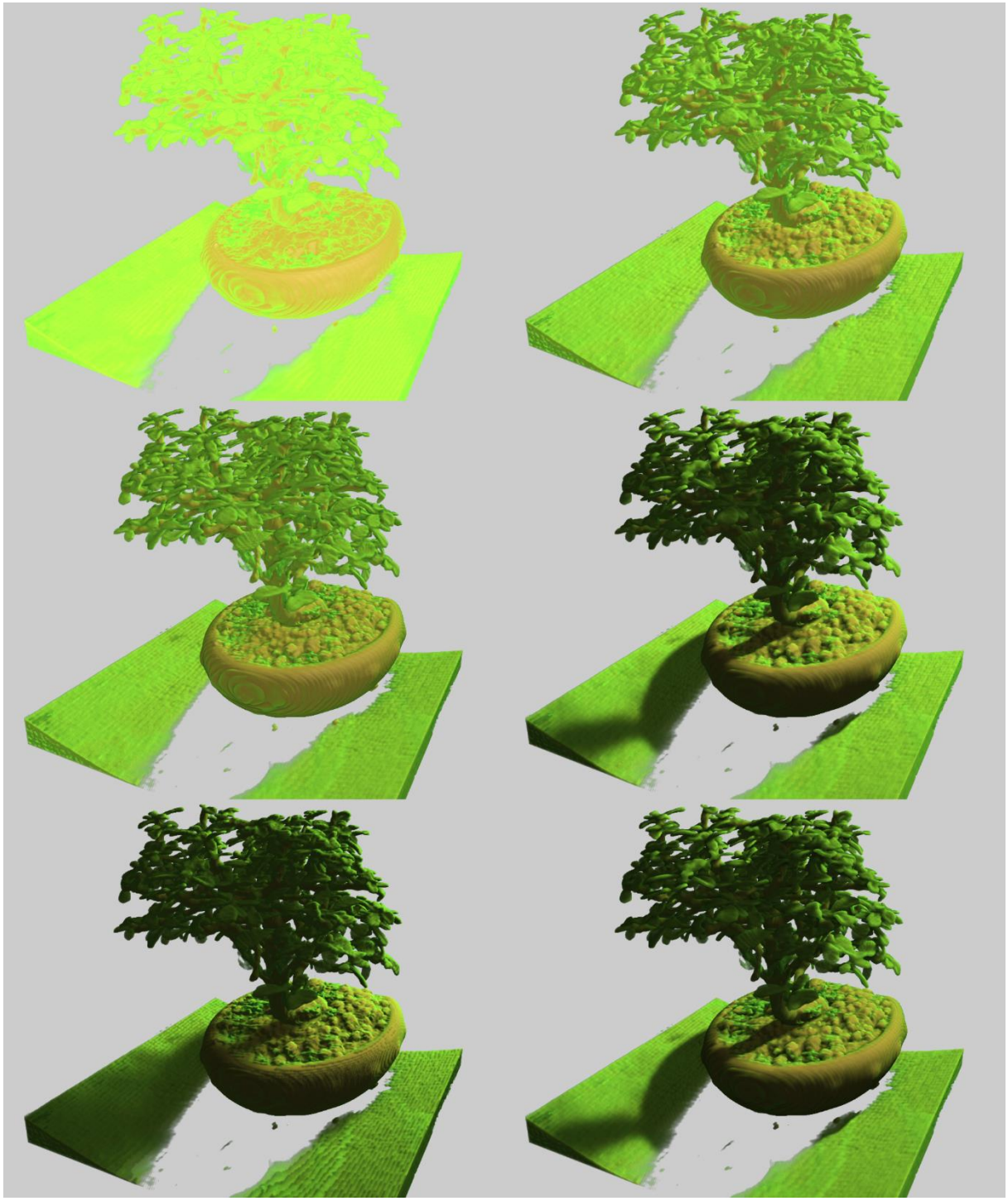
Tabulka 7 – Srovnání rychlosti vykreslování implementovaných metod globálního osvětlení. CT značí metodu trasování kužele. Předpočítání stínů neřeší AO a je proto rozlišeno trasování kužele pro výpočet stínů a výpočet stínů s AO.

I když je metoda předpočítaných stínů výpočetně rychlejší, trpí svými nedostatky. Trasování kužele umožňuje značnou parametrizaci. Nastavitelný úhel trasování kužele ke světlu α významnou měrou mění finální obraz. Možnost aplikace AO bez náročné implementace s využitím již jednou předpřipravených dat je další znatelná výhoda. Úhel ke světlu v metodě předpočítaných stínů lze ovlivnit pomocí zvoleného filtru. Ukazuje se, že pro Gaussův filtr s $\sigma = 1$ vytváří metoda měkké stíny. Výsledky vykreslovaných datasetů jsou zobrazeny na obrázcích 30, 31 a 32.

Výsledky ukazují, že obě metody dokážou generovat vržené stíny a aplikovat je na vykreslovaná data. Možným rozšířením druhé metody by byla interaktivní možnost změny aplikovaného filtru a vhodně nastavitelné parametry, které by umožnily nastavit ostrost / měkkost vrženého stínu. Implementace výpočtů stínového objemu pomocí vykreslování z plátků do trojrozměrné textury je další možné rozšíření, které by mohlo přinést výraznou úsporu v době předpočítání stínů. Pro velké datasety je vhodným rozšířením využít další optimalizační metody, aby FPS u těchto datasetů neklesaly prudce oproti vykreslování menších datasetů. Stejně tak je vhodné využít další optimalizační a filtrační postprocessingové metody, které odstraní nebo utlumí výrazné hrany způsobené diskrétní voxelovou reprezentací.

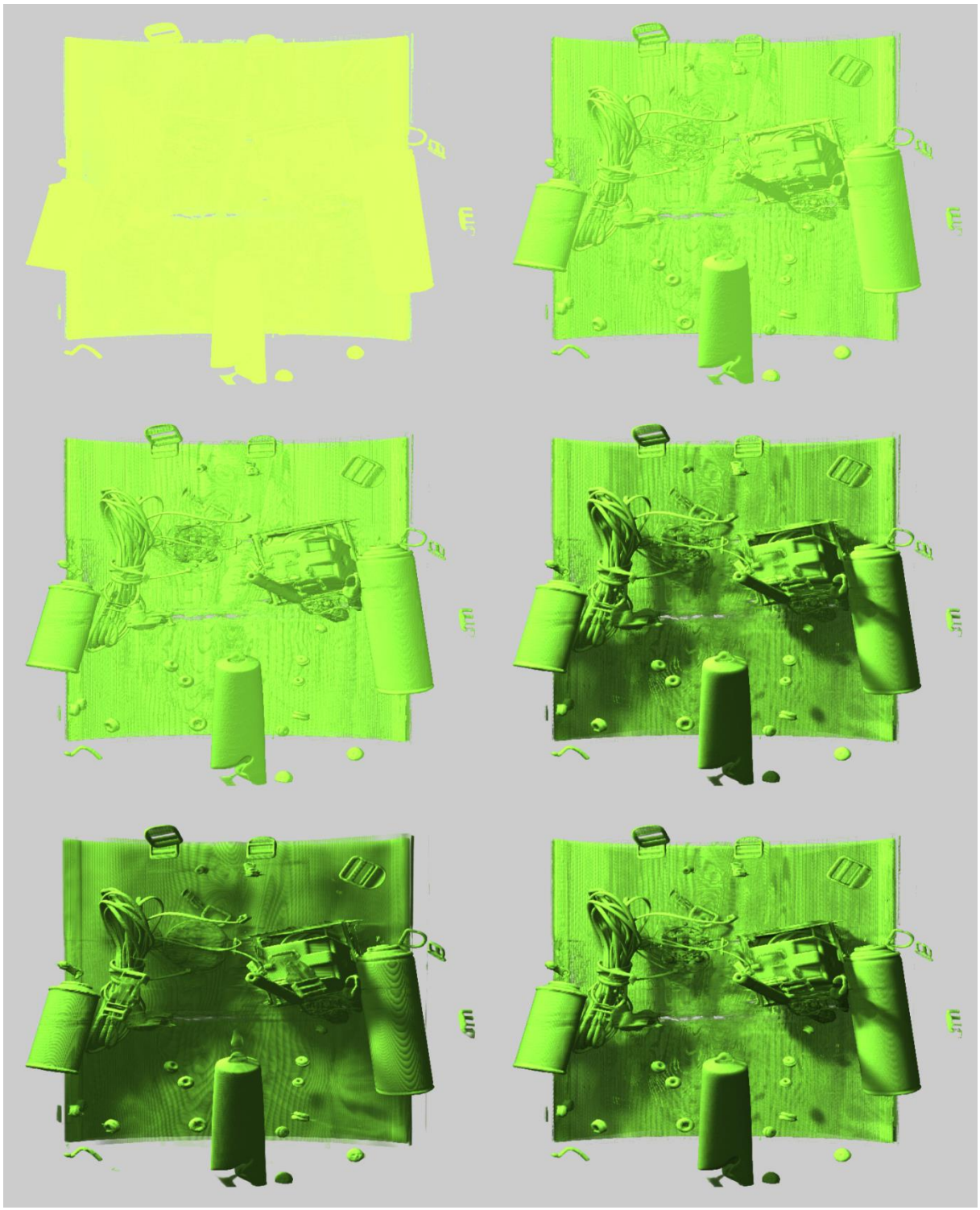


Obrázek 29 – 16 průřezů stínového objemu vygenerovaného z datasetu Bonsai. Pozice světla v ukázce je na souřadnicích $(-2{,}4;\ -2{,}8;\ 2{,}4)$, plátky jsou seřazeny zleva doprava odshora dolů od nejnižší souřadnice z po nejvyšší. První tři průřezy zachycují podložku a květináč bonsaje a na stranách je patrný stín vrhaný korunou stromu. Dalších 11 řezů zobrazuje korunu vrhající stíny. Poslední dva řezy jsou téměř bez objektů, a tedy i bez vrženého stínu.



Obrázek 30 -Vliv osvětlovacích metod na výslednou kvalitu obrazu pro dataset Bonsai. Vlevo nahoře: žádný osvětlovací model. Vpravo nahoře: lokální osvětlovací model s difúzní složkou Phongova modelu. Vlevo uprostřed: Phongovo stínování a ambient occlusion. Vpravo uprostřed: stínování a vržené stíny trasováním kužele. Vlevo dole: stínování a vržené stíny předpočítáním stínového objemu. Vpravo dole: stínování, stíny a AO metodou trasování kužele.

Obrázek 31 – Vliv osvětlovacích metod na výslednou kvalitu obrazu pro dataset Lobster. Vlevo nahoře: žádný osvětlovací model. Vpravo nahoře: lokální osvětlovací model s difúzní složkou Phongova modelu. Vlevo uprostřed: Phongovo stínování a ambient occlusion. Vpravo uprostřed: stínování a vržené stíny trasováním kužele. Vlevo dole: stínování a vržené stíny předpočítáním stínového objemu. Vpravo dole: stínování, stíny a AO metodou trasování kužele.



Obrázek 32 – Vliv osvětlovacích metod na výslednou kvalitu obrazu pro dataset Backpack. Vlevo nahoře: žádný osvětlovací model. Vpravo nahoře: lokální osvětlovací model s difúzní složkou Phongova modelu. Vlevo uprostřed: Phongovo stínování a ambient occlusion. Vpravo uprostřed: stínování a vržené stíny trasováním kužele. Vlevo dole: stínování a vržené stíny předpočítáním stínového objemu. Vpravo dole: stínování, stíny a AO metodou trasování kužele.

8 Závěr

V práci byly představeny metody výpočtu osvětlení využívané pro objemová data. Popsané osvětlovací modely umožňují realistické zobrazení voxelových dat. Vzhledem k výpočetní náročnosti těchto metod byly představeny i různé optimalizační metody, jejichž cílem je urychlit vykreslování, tak aby výsledná zobrazení dat mohlo pracovat v reálném čase. Popsané metody byly úspěšně implementovány a bylo provedeno základní srovnání obrazových výstupů a výpočetní i paměťové náročnosti jednotlivých algoritmů

Základní implementované optimalizační metody označované jako early ray termination a adaptive sampling urychlují vykreslování podle konkrétního datasetu a použité kombinace metod až čtyřnásobně. Artefakty, které by mohla implementace druhé zmíněné metody vytvářet, nebyly pozorovány.

Byly implementovány dvě různé metody globálního osvětlovacího modelu. Tyto metody leží na pomezí mezi dvěma optickými modely – mezi modelem jednoduchého rozptylu světla a vícenásobného rozptylu světla. Jednoduchý rozptyl uvažuje pouze jeden paprsek přicházející do daného bodu a jeho útlum vytváří tvrdé stíny. Oba modely však zahrnují do výpočtu stínů také okolí bodu a vytváří měkké stíny. Globální osvětlovací metody jsou také rozšířeny lokálním osvětlovacím modelem, který se také povedlo úspěšně implementovat. Byla využita difúzní složka Phongova osvětlovacího modelu, zrcadlová byla vynechána. Ačkoli lokální osvětlovací model má naivní předpoklad, že světlo do libovolného bodu uvnitř objemu dorazí bez interakce se zbytkem objemu, dokáže tento model vystihnout povrch těles lépe než aplikované globální modely. Ukazuje se, že implementovaným metodám globálního osvětlovacího modelu dodává tento model další vizuální vjem zlepšující kvalitu obrazu.

První z globálních metod předpočítává data tak, aby vhodně reprezentovala své okolí. Zobrazení probíhá trasováním kužele, který je reprezentován jen jedním paprskem procházejícím středem kužele. Vzorky na tomto paprsku jsou načítány z předpočítaného objemu dat. Trasování paprsku v kuželu je využito pro výpočet vržených stínů od zdroje světla a pro výpočet směrového ambient occlusion.

Druhá metoda je autorská. Metoda předpočítává přímo vržené stíny. Pomocí vhodné filtrace dat je stín propagován od zdroje světla skrz celý objem. Výpočet stínů probíhá pomocí čtení jedné hodnoty vrženého stínu do daného bodu.

Výsledky ukazují, že doba na výpočet předpočítaných objemů není zanedbatelná a roste se třetí mocninou velikosti. Samotné vykreslování je ale díky předpočítání méně výpočetně náročné. První metoda má kratší čas výpočtu na přípravu dat, ale nižší rychlost vykreslování, autorská metoda obráceně. Pro obě metody se podařilo vykreslovat objemová data s globálním i lokálním osvětlením v reálném čase, viz kapitola 7 Výsledky.

Možné rozšíření implementovaných metod spočívá v aplikaci dalších optimalizačních metod, které by dále zvýšily rychlost vykreslování, nebo odstranění artefaktů způsobujících strukturu (*wood grain* artefakty). Přípravná fáze předpočítaných stínů je implementovaná na

compute shaderu. Další rozšíření práce může být implementace tohoto výpočtu vykreslováním do plátků trojrozměrné textury, které je použité v přípravné fázi první metody.

9 Zdroje

[1] A. V. OPPENHEIM a R. W. SCHAFER, 1975. *Digital Signal Processing*. ISBN 978-0-13-214635-7.

[2] ALEXEY PANTELEEV, 2016. VXAO: Voxel Ambient Occlusion. *NVIDIA Developer* [online]. [vid. 2020-10-01]. Dostupné z: https://developer.nvidia.com/vxao-voxel-ambient-occlusion

[3] ANDREW S. GLASSNER, 1995. Signals and systems. In: *Principles of digital image synthesis*. B.m.: Elsevier. ISBN 978-0-12-286251-9.

[4] BAVOIL, Louis, Miguel SAINZ a Rouslan DIMITROV, 2008. Image-space horizon-based ambient occlusion. In: *ACM SIGGRAPH 2008 talks* [online]. New York, NY, USA: Association for Computing Machinery, s. 1 [vid. 2020-10-01]. SIGGRAPH '08. ISBN 978-1-60558-343-3. Dostupné z: doi:10.1145/1401032.1401061

[5] BERGER, Christoph, 2003. A Framework for Flexible, Hardware-Accelerated, and High-Quality Volume Rendering. *A Framework for Flexible, Hardware-Accelerated, and High-Quality Volume Rendering* [online] [vid. 2020-02-07]. Dostupné z: http://old.cescg.org/CESCG-2003/CBerger/index.html

[6] BLINN, James F., 1994. Jim Blinn's Corner: Compositing, Part I: Theory. *IEEE Computer Graphics and Applications*. **Září 1994**.

[7] BOKŠANSKÝ, Jakub, Adam POSPÍŠIL a Jiří BITTNER, 2017. *VAO++: Practical Volumetric Ambient Occlusion for Games* [online]. B.m.: The Eurographics Association [vid. 2020-11-15]. ISBN 978-3-03868-045-1. Dostupné z: doi:10.2312/sre.20171192

[8] CAMPAGNOLO, Leonardo Q. a Waldemar CELES, 2019. Interactive directional ambient occlusion and shadow computations for volume ray casting. *Computers & Graphics* [online]. **84**, 66–76. ISSN 0097-8493. Dostupné z: doi:10.1016/j.cag.2019.08.009

[9] DECAUDIN, Philippe a Fabrice NEYRET, 2009. Volumetric Billboards. *Computer Graphics Forum* [online]. **28**(8), 2079–2089. Dostupné z: doi:10.1111/j.1467-8659.2009.01354.x

[10] DÍAZ, Jose, Timo ROPINSKI, Isabel NAVAZO, Enrico GOBBETTI a Pere-Pau VÁZQUEZ, 2017. An experimental study on the effects of shading in 3D perception of volumetric models. *The Visual Computer* [online]. **33**(1), 47–61. ISSN 0178-2789, 1432-2315. Dostupné z: doi:10.1007/s00371-015-1151-6

[11] FAVERA, E. C. D. a W. CELES, 2012. Ambient Occlusion Using Cone Tracing with Scene Voxelization. In: *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*: *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images* [online]. s. 142–149. Dostupné z: doi:10.1109/SIBGRAPI.2012.28

[12] HADWIGER, Markus, nedatováno. Volume Rendering - Introduction. In: [online]. King Abdullah University of Science and Technology. Dostupné z: http://isgwww.cs.uni-magdeburg.de/miccai-vis/markus_hadwiger.pdf

[13] HADWIGER, Markus, Joe M. KNISS, Christof REZK-SALAMA, Daniel WEISKOPF a Klaus ENGEL, 2006. *Real-time Volume Graphics*. Natick, MA, USA: A. K. Peters, Ltd. ISBN 978-1-56881-266-3.

[14] JIMENEZ, Jorge, Xian-Chun WU, Angelo PESCE a Adrian JARABO, nedatováno. Practical Realtime Strategies for Accurate Indirect Occlusion. 9.

[15] JOHN PAWASAUSKAS, 1997. Volume Rendering With Ray Casting. *Volume Rendering With Ray Casting* [online] [vid. 2020-08-28]. Dostupné z: http://web.cs.wpi.edu/~matt/courses/cs563/talks/powwie/p1/ray-cast.htm

[16]KNITTEL, Guenter, 2002. Using pre-integrated Transfer Functions in an Interactive Software System for Volume [online]. [vid. 2020-11-15]. ISSN 1017-4656. Dostupné z: doi:10.2312/egs.20021047

[17]KRUGER, J. a R. WESTERMANN, 2003. Acceleration techniques for GPU-based volume rendering. In: *IEEE Visualization 2003*: *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control* [online]. Seattle, WA, USA: IEEE, s. 287–292 [vid. 2020-09-20]. ISBN 978-0-7803-8120-9. Dostupné z: doi:10.1109/VISUAL.2003.1250384

[18]LEE, Jong a Timothy NEWMAN, 2007. New Method for Opacity Correction in Oversampled Volume Ray Casting.

[19]LESAR, Žiga, Ciril BOHAK a Matija MAROLT, 2018. Real-time interactive platform-agnostic volumetric path tracing in webGL 2.0. In: *the 23rd International ACM Conference*: *Proceedings of the 23rd International ACM Conference on 3D Web Technology - Web3D '18* [online]. Poznań, Poland: ACM Press, s. 1–7 [vid. 2020-08-10]. ISBN 978-1-4503-5800-2. Dostupné z: doi:10.1145/3208806.3208814

[20]MARIUS GAVRILESCU, 2011. *VISUALIZATION AND GRAPHICAL PROCESSING OF VOLUME DATA*. B.m. Universitatea Tehnică Invatator Gheorghe Asachi.

[21]MARTIN KRAUS a THOMAS ERTL, 2005. Pre-Integrated Volume Rendering. In: *The visualization handbook*. Amsterdam ; Boston: Elsevier-Butterworth Heinemann. ISBN 978-0-12-387582-2.

[22]MATT PHARR a SIMON GREEN, 2007. Chapter 17. Ambient Occlusio. In: *GPU gems* [online]. B.m.: NVIDIA Developer [vid. 2020-11-15]. Dostupné z: https://developer.download.nvidia.com/books/HTML/gpugems/gpugems_ch17.html

[23]MAX, N., 1995. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics* [online]. **1**(2), 99–108. ISSN 1941-0506. Dostupné z: doi:10.1109/2945.468400

[24]MAX, Nelson a Min CHEN, 2010. Local and Global Illumination in the Volume Rendering Integral [online]. 16 pages. Dostupné z: doi:10.4230/DFU.SCIVIZ.2010.259

[25]MILAN IKITS, JOE KNISS, AARON LEFOHN a CHARLES HANSEN, 2007. Chapter 39. Volume Rendering Techniques. In: *GPU gems* [online]. B.m.: NVIDIA Developer [vid. 2020-08-10]. Dostupné z: https://developer.nvidia.com/gpugems/gpugems/part-vi-beyond-triangles/chapter-39-volume-rendering-techniques

[26]MITTRING, Martin, 2007. Finding next gen: CryEngine 2. In: *ACM SIGGRAPH 2007 courses*: *ACM SIGGRAPH 2007 courses on - SIGGRAPH '07* [online]. San Diego, California: ACM Press, s. 97 [vid. 2020-10-01]. ISBN 978-1-4503-1823-5. Dostupné z: doi:10.1145/1281500.1281671

[27]NYQUIST, H., 1928. Certain Topics in Telegraph Transmission Theory. *Transactions of the American Institute of Electrical Engineers* [online]. **47**(2), 617–644. ISSN 2330-9431. Dostupné z: doi:10.1109/T-AIEE.1928.5055024

[28]PORTER, Thomas a Tom DUFF, 1984. Compositing digital images. In: Proceedings of the 11th annual conference on Computer graphics and interactive techniques [online]. New York, NY, USA: Association for Computing Machinery, s. 253–259 [vid. 2020-09-16]. SIGGRAPH '84. ISBN 978-0-89791-138-2. Dostupné z: doi:10.1145/800031.808606

[29]ROETTGER, Stefan, Stefan GUTHE, Daniel WEISKOPF, Thomas ERTL a Wolfgang STRASSER, 2003. Smart hardware-accelerated volume rendering. In: *Proceedings of the symposium on Data visualisation 2003*. Goslar, DEU: Eurographics Association, s. 231–238. VISSYM '03. ISBN 978-1-58113-698-2.

[30]SERGEY FOMEL, 2001. *Forward interpolation* [online]. B.m. [vid. 2020-09-12]. b.n. Dostupné z: http://ahay.org/RSF/book/sep/forwd/paper_html/node10.html

[31]STEFAN GUTHE a PAUL HECKBERT, 2003. *Non-Power-of-Two Mipmap Creation* [online]. 2003. B.m.: NVIDIA Corporation. Dostupné z: http://download.nvidia.com/developer/Papers/2005/NP2_Mipmapping/NP2_Mipmap_Creation.pdf

[32]SZIRMAY-KALOS, L., T. UMENHOFFER, B. TOTH, L. SZECSI a M. SBERT, 2010. Volumetric Ambient Occlusion for Real-Time Rendering and Games. *IEEE Computer Graphics and Applications* [online]. **30**(1), 70–79. ISSN 0272-1716. Dostupné z: doi:10.1109/MCG.2010.19

[33]YALÇINER, Bora a Yusuf SAHILLIOĞLU, 2020. Voxel transformation: scalable scene geometry discretization for global illumination. *Journal of Real-Time Image Processing* [online]. **17**(5), 1585–1596. ISSN 1861-8200, 1861-8219. Dostupné z: doi:10.1007/s11554-019-00919-1

[34]ZHUKOV, S., A. IONES a G. KRONIN, 1998. An ambient light illumination model. In: George DRETTAKIS a Nelson MAX, ed. *Rendering Techniques '98* [online]. Vienna: Springer, s. 45–55. Eurographics. ISBN 978-3-7091-6453-2. Dostupné z: doi:10.1007/978-3-7091-6453-2_5

Použité jazyky a knihovny a jejich specifikace

- Java – https://docs.oracle.com/javase/specs/jls/se15/jls15.pdf
- OpenGL – https://www.khronos.org/registry/OpenGL/specs/gl/glspec46.core.pdf
- GLSL – https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.40.pdf
- LWJGL – https://www.lwjgl.org/

Využité datasety a jejich autoři

- Backpack – Kevin Kreeger, Viatronix Inc., USA
- Bonsai – S. Roettger, VIS, University of Stuttgart
- Lobster – VolVis distribution of SUNY Stony Brook, NY, USA

Zdroj datasetů: *Open Scientific Visualization Datasets* [online] [vid. 2020-08-28]. Dostupné z: https://klacansky.com/open-scivis-datasets/

```
#version 430
#extension GL_ARB_compute_shader : enable
// počet spuštění programu pro jednotlivý voxel
layout (local_size_x = 1, local_size_y = 1, local_size_z = 1 ) in;
// vstupní a výstupní obrázky - zdroj dat a úložiště
layout (r32f, binding = 0) uniform readonly image3D source;
layout (r32f, binding = 1) uniform image3D shadows; //readwrite
// textura přechodové funkce a filtru
uniform sampler1D tf;
uniform sampler1D tFilter;

uniform vec3 lightPos; // pozice světla SSS
uniform float voxelSize; // velikost voxelu
uniform ivec3 sliceDir; // směr propagace světla
uniform int sliceMax; // počet plátků ve směru propagace
uniform int sliceC; // absolutní pozice aktuálního plátku
uniform mat3 modelMat; // modelovací matic - SSM → SSS
uniform mat3 inverseModelMat; // inverzní modelovací matice SSS → SSM

// funkce na přepočet souřadnic z absolutních celočíselných do relativních
vec3 toRelative(ivec3 v, sampler3D tex, int lod){
    ivec3 dim = textureSize(tex, lod);
    vec3 off = vec3(1.0)/(2.0*dim);
    vec3 step = vec3(1.0)/dim;
    vec3 rel = off + v*step;
    return rel;
}

void main(){
    ivec3 absSliceDir = abs(sliceDir); // pomocný vektor - osa propagace
    ivec3 invAbsSliceDir = 1-absSliceDir; // pomocný vektor
    // pozice aktuálně počítaného voxelu
    ivec3 sourceCoord =
        ivec3(gl_WorkGroupID)*invAbsSliceDir + sliceC*absSliceDir;
    // velikost objemu ve voxelech
    ivec3 dim =
        ivec3(gl_NumWorkGroups)*invAbsSliceDir + sliceMax*absSliceDir;
    // hodnota aktuálního voxelu a aplikace přechodové funkce
    float value = imageLoad(source, sourceCoord).r;
    value = texture(tf, value).a;
    // převod z absolutních souřadnic na interval (0;1)
    vec3 relativeCoordsMS = toRelative(sourceCoord, dim);
    // převod souřadnic voxelu z SSM do SSS
    vec3 relativeCoordsWS = modelMat * relativeCoordsMS;
    // výpočet vektoru ke světlu
    vec3 lightDirWS = normalize(lightPos - relativeCoordsWS);
    // násobek vektoru tak, aby vektor na ose plátkování měl hodnotu 1
    float k = abs(voxelSize/dot(absSliceDir, lightDirWS));
    // souřadnice předchozí úrovně ve směru paprsku
    vec3 prevLevelRelativeCoordsWS = relativeCoordsWS + k * lightDirWS;
    // převod zpět na interval (0;1)
    vec3 prevLevelRelativeCoordsMS =
        inverseModelMat * prevLevelRelativeCoordsWS;
    // pomocné proměnné na přepočet souřadnic z relativních do absolutních
    vec3 toAbsOff = vec3(1.0)/(2.0*dim);
    vec3 toAbsStep = vec3(1.0)/dim;
```

```
    // výpočet souřadnich průsečíku předchozí roviny plátkování a paprsku
    // druhý člen přičítá ve směru plátkování 0.5 kvůli chybě z nepřesnosti
    ivec3 prevLevelAbsCoordsRoundDown =
        ivec3((prevLevelRelativeCoordsMS-toAbsOff)*dim +
        0.5 * absSliceDir *
        sign(dot(absSliceDir, prevLevelRelativeCoordsMS)));
    // hodnoty určující souřadnice hraničních bodů B
    ivec3 cycleLimitsMin = invAbsSliceDir*(-3) +
        prevLevelAbsCoordsRoundDown;
    ivec3 cycleLimitsMax = invAbsSliceDir*  4  +
        prevLevelAbsCoordsRoundDown;
    // pomocná hodnota pro výpočet interpolačního parametru
    vec3 floatPrevLevelAbsCoords =
        (prevLevelRelativeCoordsMS - toAbsOff)*dim;
    // interpolační parametr - desetinná část vzdálenosti mezi body P a B
    vec3 t = invAbsSliceDir *
        (floatPrevLevelAbsCoords - prevLevelAbsCoordsRoundDown);
    // přepočet hodnot z <0,1> na <1/2d,1-1/2d>
    int fDim = textureSize(tFilter, 0);
    t = (t*(fDim-1.0)+0.5)/fDim;
    // načtení 6 hodnot textury pro 3 souřadnicové osy s param. t a 1-t
    vec4 tx  = texture(tFilter, t.x);vec4 txi = texture(tFilter, 1-t.x);
    vec4 ty  = texture(tFilter, t.y);vec4 tyi = texture(tFilter, 1-t.y);
    vec4 tz  = texture(tFilter, t.z);vec4 tzi = texture(tFilter, 1-t.z);
    // seřazení parametrů pro další výpočty
    vec3[8] c;
    c[0] = vec3( tx.a,  ty.a,  tz.a);c[1] = vec3( tx.b,  ty.b,  tz.b);
    c[2] = vec3( tx.g,  ty.g,  tz.g);c[3] = vec3( tx.r,  ty.r,  tz.r);
    c[4] = vec3(txi.r, tyi.r, tzi.r);c[5] = vec3(txi.g, tyi.g, tzi.g);
    c[6] = vec3(txi.b, tyi.b, tzi.b);c[7] = vec3(txi.a, tyi.a, tzi.a);
    // výstupní hodnota intenzity s aplikovaným filtrem
    float outputValue = 0.0;
    // iterace přes všechny body B
    for(int ix = cycleLimitsMin.x; ix <= cycleLimitsMax.x; ix++){
        for(int iy = cycleLimitsMin.y; iy <= cycleLimitsMax.y; iy++){
            for(int iz = cycleLimitsMin.z; iz <= cycleLimitsMax.z; iz++){
                outputValue +=
                imageLoad(shadows, ivec3(ix, iy, iz)).r *
                c[ix - cycleLimitsMin.x + absSliceDir.x*3].x *
                c[iy - cycleLimitsMin.y + absSliceDir.y*3].y *
                c[iz - cycleLimitsMin.z + absSliceDir.z*3].z *
                // negace koeficientu pro nevyužitou osu = 1/g(0)
                2.50662827463;
            }
        }
    }
    // výsledná hodnota je součet zastínění aktuálního bodu A a bodů B
    float saveValue = min(outputValue + value, 1);
    // uložení hodnoty na aktuální pozici
    imageStore(shadows, sourceCoord, vec4(saveValue, 0.0, 0.0, 1.0));
}
```

Ukázka 13 - Implementace výpočtu předpočítaných stínů na compute shaderu.

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor:	**Bc. Ondřej Caha**
Studium:	I1800769
Studijní program:	N1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název diplomové práce:	**Osvětlení při vykreslování volumetrických modelů**
Název diplomové práce AJ:	Lighning techniques for volume rendering

Cíl, metody, literatura, předpoklady:

Cíl práce:

Prozkoumat přístupy a techniky používané pro osvětlení volumetrických modelů a jejich využití na hardwarově akcelerovaných grafických kartách.

Postup prací:

1. Prozkoumat principy a metody používané pro osvětlení volumetrických dat
2. Vytvořit přehled používaných přístupů a algoritmů pro vizualizaci konkrétních volumetrických modelů
3. Pro vybranou metodu navrhnout řešení s využitím programovatelných grafických karet
4. Navržené řešení implementovat a otestovat
5. Zhodnotit dosažené výsledky

- CAMPAGNOLO, Leonardo Q. a Waldemar CELES, 2019. Interactive directional ambient occlusion and shadow computations for volume ray casting. Computers {\& Graphics [online]. 84, 66?76. ISSN 0097-8493. Dostupné z: doi:10.1016/j.cag.2019.08.009
- ANON., nedatováno. Optical models for direct volume rendering - IEEE Journals {\& Magazine [online] [vid. 2019-10-04]. Dostupné z: https://ieeexplore.ieee.org/abstract/document/468400
- HADWIGER, Markus, Joe M. KNISS, Christof REZK-SALAMA, Daniel WEISKOPF a Klaus ENGEL, 2006. Real-time Volume Graphics. Natick, MA, USA: A. K. Peters, Ltd. ISBN 978-1-56881-266-3.
- FAVERA, E. C. D. a W. CELES, 2012. Ambient Occlusion Using Cone Tracing with Scene Voxelization. In: 2012 25th SIBGRAPI Conference on Graphics, Patterns and Images: 2012 25th SIBGRAPI Conference on Graphics, Patterns and Images [online]. s. 142?149. Dostupné z: doi:10.1109/SIBGRAPI.2012.28

Garantující pracoviště:	Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu
Vedoucí práce:	Ing. Bruno Ježek, Ph.D.
Oponent:	prof. RNDr. PhDr. Antonín Slabý, CSc.

Datum zadání závěrečné práce: 4.5.2020