

VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Aplikovaná informatika

PŘEDPOČÍTANÉ STATICKÉ OSVĚTLENÍ
A GENEROVÁNÍ LIGHTMAP VE 3D SCÉNÁCH

Bakalářská práce

Autor práce: Filip Vencelides

Vedoucí práce: doc. Ing. Zbyněk Bureš, Ph.D.

Jihlava 2025

Vysoká škola polytechnická Jihlava

Tolstého 16, 586 01 Jihlava

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Autor práce: **Filip Vencelides**

Studijní program: Aplikovaná informatika

Garant studijního programu: Ing. Lenka Kuklišová Pavelková, Ph.D.

Název práce: **Předpočítané statické osvětlení a generování lightmap ve 3D scénách**

Vedoucí práce: doc. Ing. Zbyněk Bureš, Ph.D.

Cíl práce: Seznamte se s problematikou a možnostmi předpočítání statického osvětlení do podoby tzv. lightmap a prostudujte existující metody řešení. Proveďte rozbor jednotlivých fází procesu generování lightmap a implementujte aplikační software, který umožní výpočet statického osvětlení pro uživatelem zadanou 3D scénu, definovanou geometrií a zdroji světla. Výstupem nástroje pak bude 3D scéna, obohacená o vygenerované lightmapy a přidružené souřadnice UV mapování, podle kterých lze lightmapy aplikovat na geometrii.

Abstrakt

Bakalářská práce se věnuje průzkumu metod pro předpočítání globálního difúzního osvětlení ve virtuálních 3D scénách, se zvláštním zaměřením na jeho reprezentaci prostřednictvím lightmap. Pro demonstraci celého procesu byl navržen a implementován softwarový nástroj, jehož stěžejní části implementace týkající se generování lightmap jsou podrobně popsány v textu.

Nástroj umožňuje interaktivní 3D zobrazení scény s možností konfigurace světelných zdrojů a nastavení parametrů pro výpočet lightmap. Klíčovou funkcí nástroje pak je samotné generování lightmap, které probíhá progresivně za použití metody path tracingu, realizovaného na CPU. Na závěr jsou prezentovány výsledky práce ve formě grafických výstupů vytvořených nástrojem, přičemž je zhodnocena funkčnost celého řešení spolu s návrhy na jeho další rozvoj.

Klíčová slova

předpočítané osvětlení, globální osvětlení, nepřímé osvětlení, lightmap, path tracing, kvazi-Monte Carlo integrování, posloupnost s nízkou diskrepancí, Hammersleyho posloupnost, Haltonova posloupnost, sférické harmonické funkce, RGBM kódování

Abstract

This bachelor thesis deals with techniques for precomputed global diffuse illumination in virtual 3D environments, with a specific focus on its representation using lightmaps. A software tool was designed and implemented to demonstrate the entire process. The key parts of the implementation related to lightmap generation are described in detail in the text.

The tool allows to visualize a 3D scene in an interactive way, including the configuration of light sources and the adjustment of parameters used for the lightmap computation. The core functionality of the tool is the progressive generation of lightmaps using path tracing implemented on the CPU. Finally, the results are presented in the form of graphical outputs generated by the tool, and the overall functionality of the solution is evaluated along with suggestions for future improvements.

Keywords

precomputed lighting, global illumination, indirect lighting, lightmap, path tracing, quasi-Monte Carlo integration, low-discrepancy sequence, Hammersley sequence, Halton sequence, spherical harmonics, RGBM encoding

Prohlašuji, že předložená bakalářská práce je původní a zpracoval/a jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil/a autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „AZ“).

Byl/a jsem seznámen/a s tím, že na mou bakalářskou práci se plně vztahuje **AZ**, zejména § 60 (školní dílo).

Podle § 47b zákona o vysokých školách souhlasím se zveřejněním své práce podle Směrnice pro vedení, vypracování a zveřejňování závěrečných prací na VŠPJ, a to bez ohledu na výsledek obhajoby.

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom/a toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mě požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutím licence.

V Jihlavě dne 14. dubna 2025

.....

Podpis studenta

Poděkování

Děkuji vedoucímu práce panu doc. Ing. Zbyňku Burešovi, Ph.D. za jeho odborné vedení, rady a trpělivost, se kterou mě vedl při psaní této práce.

Velmi si vážím také podpory, vstřícnosti a pochopení, kterých se mi od rodiny, přátel a kolegů z práce dostávalo po celou dobu studia na Vysoké škole polytechnické Jihlava.

Obsah

Seznam obrázků.....	7
Seznam zkratk.....	10
Úvod	11
1 Teoretický základ pro předpočítané osvětlení.....	12
1.1 Světlo v počítačové grafice	12
1.2 Předpočítané globální osvětlení	18
1.3 Lightmapping	20
1.4 Veličiny a formáty předpočítaného osvětlení.....	27
1.5 Path tracing.....	34
2 Návrh řešení.....	41
2.1 Požadované funkce nástroje.....	42
2.2 Reprezentace scény	43
2.3 Použité technologie	47
3 Tvorba lightmapového atlasu.....	50
4 Generování lightmap	54
4.1 Akcelerační struktury pro výpočet osvětlení	54
4.2 Preprocessing	58
4.3 Výpočet osvětlení	66
4.4 Postprocessing.....	74
5 Souborový formát scény	77
5.1 Souborový formát lightmap.....	77
6 Vykreslovací 3D engine	79
6.1 Zobrazovací módy.....	79
7 Výsledky.....	82
7.1 Znamá omezení	88
Závěr	90
Seznam použité literatury	93

Seznam obrázků

Obr. 1: Stíny vytvořené bezrozměrným bodovým zdrojem světla (nahore) a plošným zdrojem světla (dole)	13
Obr. 2: Světelný zdroj osvětluje povrch, přičemž normála n určuje hemisféru všech možných směrů dopadajícího světla – vlevo jako bezrozměrný světelný zdroj, vpravo jako plošný světelný zdroj.	14
Obr. 3: Různé typy světelných zdrojů: nahore bodový a směrový, dole reflektorový a plošný .	15
Obr. 4: Různé typy BRDF	16
Obr. 5: Použití lightmap ve videohře Quake (1996) od společnosti id Software: (a) albedo, (b) lightmapa s nearest-neighbor filtrováním, (c) lightmapa s bilineárním filtrováním, (d) kombinace albeda a bilineárně filtrované lightmapy	21
Obr. 6: Příklad textury lightmapy zobrazené v texturovém prostoru	21
Obr. 7: Porovnání povrchu osvětleného stejnou lightmapou za využití bilineárního filtrování (vlevo) a bikubického filtrování (vpravo)	22
Obr. 8: Různé úrovně, na kterých je osvětlení počítáno: (a) ploché stínování: jednou pro celý polygon, (b) Gouraudovo stínování: pro každý vrchol s interpolací mezi nimi, (c) Phongovo stínování: pro každý pixel výsledného obrazu	24
Obr. 9: Irradiance volume s pravidelně rozmístěnými light probes, znázorněnými pomocí koulí.	25
Obr. 10: Nepravidelně rozmístěné light probes (znázorněné jako koule) propojené do sítě tvořené čtyřstěny pomocí tetraedrizaace – snímek z prostředí herního enginu Unity zobrazující scénu videohry Shadowgun od společnosti MADFINGER Games.....	26
Obr. 11: Environment mapa „Grace Cathedral“ uchováající zář (vlevo) a jí odpovídající irradiance environment mapa uchováající intenzitu záření reprezentovaná pomocí textury (uprostřed) a pomocí sférických harmonických funkcí třetího řádu (vpravo)	31
Obr. 12: Tři způsoby vzorkování ve 2D: (a) náhodné vzorkování, (b) pravidelně stratifikované vzorkování, (c) stratifikované vzorkování	37
Obr. 13: Několik prvních bodů posloupností s nízkou diskrepancí ve 2D: (a) Haltonovy posloupnosti – 216 bodů a (b) Hammersleyho posloupnosti – 256 bodů.....	38
Obr. 14: Povrch geometrie po rozvinutí do lightmapového atlasu – každý povrch je obarven podle příslušnosti ke konkrétnímu chartu v lightmapovém atlasu (viz Obr. 15). Políčka šachovnicového vzoru na površích geometrie odpovídají velikosti jednoho texelu v lightmapovém atlasu.....	50
Obr. 15: Povrch geometrie parametrizovaný a uspořádaný do lightmapového atlasu. Jednotlivé charty jsou barevně odlišeny.	51
Obr. 16: Zdrojový kód akcelerační struktury pro efektivnější výpočet barycentrických souřadnic ve 2D prostoru	56
Obr. 17: Zjednodušené schéma 2D průřezu light grid zobrazující přiřazení světelných zdrojů jednotlivým buňkám mřížky – bodový zdroj (žlutě) a reflektorový zdroj (zeleně)	58
Obr. 18: Grafické znázornění vytváření vzorků na povrchu geometrie pomocí posloupnosti s nízkou diskrepancí – zobrazení v UV prostoru a 3D prostoru	60
Obr. 19: Princip omezení čtvercového polygonu texelu (zeleně) trojúhelníkovým polygonem (modře) a výsledný průnik (oranžově) s jeho geometrickým středem (červeně).....	60

Obr. 20: Porovnání výstupu standardní rasterizace (vlevo) a konzervativní rasterizace (vpravo)	62
Obr. 21: Předdefinované posuny v UV prostoru, jež jsou násobeny polovinou velikosti texelu a aplikovány v několika rasterizačních průchodech za účelem emulace konzervativní rasterizace	62
Obr. 22: Artefakty vzniklé v důsledku vzorkování na hranách trojúhelníků – v rozích místnosti dochází k nesprávnému osvětlení, které vzniká v důsledku neurčitosti, které straně povrchu daný vzorek přísluší. Tento problém se typicky vyskytuje na rozhraní dvou oblastí s rozdílným osvětlením, kde by povrch měl tvořit světelnou bariéru. Vzorky umístěné přesně na hraně však mohou vést k úniku světla mezi těmito oblastmi, což se projevuje jako nepřírozeně ostré přechody na hranách geometrie.	63
Obr. 23: Projev artefaktů způsobených únikem světla do interiéru uzavřené místnosti.	65
Obr. 24: Společné rozhraní akumulátoru. Konkrétní specializace se liší podle formátu osvětlení v lightmapě.	69
Obr. 25: Kód sloužící pro projekci příchozí záře v daném směru do reprezentace předpočítaného osvětlení pomocí SH třetího řádu, používané nástrojem. Samotná zář není v této části kódu zohledněna, její projekce však odpovídá násobení záře s výsledkem vráceným touto metodou.	70
Obr. 26: Kód ve fragment shaderu, který při vykreslování slouží pro získání intenzity záření v daném směru povrchové normály. Kód provádí skalární součin koeficientů dané povrchové normály s koeficienty SH třetího řádu obsahujícími předkonvolovanou intenzitu záření zahrnujícími i koeficienty pro projekci povrchové normály na bázové funkce SH. Koeficienty předkonvolované intenzity záření jsou získány vzorkováním lightmapy v příslušném místě povrchu.	71
Obr. 27: Lightmapa na povrchu objektu před dilatací (vlevo) a po dilataci (vpravo). Povšimněte si chybějícího osvětlení podél hran v místě švů (tmavé hrany). U spodních snímků byl pro lepší viditelnost upraven kontrast.	75
Obr. 28: Průběžně generovaná lightmap bez využití denoisingu.	76
Obr. 29: Průběžně generovaná lightmapa s využitím denoisingu.	76
Obr. 30: Vizualizace příslušnosti povrchů k jednotlivým chartům v rámci lightmapového atlasu. Zobrazení švů chartů je aktivní.	80
Obr. 31: Vizualizace hustoty texelů v rámci lightmapového atlasu. Barevné označení povrchů vychází z poměru velikosti texelu vůči ploše, kterou pokrývá – modře nedostatečné, zeleně optimální a červeně nadměrné rozlišení. Zobrazení švů chartů je aktivní.	81
Obr. 32: Scéna osvětlená pomocí dynamicky počítaného přímého osvětlení. Povrch je pokryt barvou albedo, při výpočtu osvětlení je využíváno normály z normálové mapy.	82
Obr. 33: Scéna osvětlená pomocí dynamicky počítaného přímého osvětlení. Na povrchu je zobrazeno pouze osvětlení, při výpočtu osvětlení je využíváno normály z normálové mapy.	83
Obr. 34: Scéna osvětlená pomocí dynamicky počítaného přímého osvětlení. Na povrchu je zobrazeno pouze osvětlení, při výpočtu osvětlení je využíváno povrchové normály.	83
Obr. 35: Scéna osvětlená pomocí předpočítaného globálního osvětlení ze směrové lightmapy ve formátu SH třetího řádu. Povrch je pokryt barvou albedo, při rekonstrukci osvětlení z lightmapy je využíváno normály z normálové mapy.	84

Obr. 36: Scéna osvětlená pomocí předpočítaného globálního osvětlení ze směrové lightmapy ve formátu SH třetího řádu. Na povrchu je zobrazeno pouze osvětlení, při rekonstrukci osvětlení z lightmapy je využíváno normály z normálové mapy.	84
Obr. 37: Scéna osvětlená pomocí předpočítaného globálního osvětlení z nesměrové lightmapy. Povrch je pokryt barvou albeda, při rekonstrukci osvětlení z lightmapy je implicitně využíváno povrchové normály.	85
Obr. 38: Scéna osvětlená pomocí předpočítaného globálního osvětlení z nesměrové lightmapy. Na povrchu je zobrazeno pouze osvětlení, při rekonstrukci osvětlení z lightmapy je implicitně využíváno povrchové normály.	85
Obr. 39: Scéna osvětlená pomocí předpočítaného přímého osvětlení ze směrové lightmapy ve formátu SH třetího řádu. Na povrchu je zobrazeno pouze osvětlení, při rekonstrukci osvětlení z lightmapy je využíváno normály z normálové mapy.	86
Obr. 40: Atrium scény Sponza (autor: Frank Meinl) osvětlené pomocí předpočítaného globálního osvětlení ze směrové lightmapy ve formátu SH třetího řádu. Horní část zobrazuje scénu s plnými materiály a texturami, zatímco spodní část zobrazuje pouze osvětlení.	87
Obr. 41: Postranní část atria scény Sponza (autor: Frank Meinl) osvětlené pomocí předpočítaného globálního osvětlení ze směrové lightmapy ve formátu SH třetího řádu. Horní část zobrazuje scénu s plnými materiály a texturami, zatímco spodní část zobrazuje pouze osvětlení.	88
Obr. 42: Nespojivosti osvětlení ve švech chartů. Levá část zobrazuje rozdělení povrchu do jednotlivých chartů (barevně odlišené oblasti), zatímco pravá část ukazuje výsledné osvětlení po vygenerování lightmapy. Ve švech mezi charty jsou patrné ostré přechody v osvětlení.	89

Seznam zkratek

AABB	<i>axis-aligned bounding box</i> – čtverec/kvádr zarovnaný s osami souřadného systému, používaný pro ohraničení množiny bodů v prostoru
BRDF	<i>bidirectional reflectance distribution function</i> – obousměrná distribuční funkce odrazu
BSDF/BxDF	<i>bidirectional scattering distribution function</i> – obousměrná distribuční funkce rozptylu
BTDF	<i>bidirectional transmittance distribution function</i> – obousměrná distribuční funkce lomu
HDR	<i>high dynamic range</i> – vysoký dynamický rozsah
MIS	<i>multiple importance sampling</i>
NEE	<i>next event estimation</i>
PDF	<i>probability density function</i> – hustota pravděpodobnosti
SDR	<i>standard dynamic range</i> – standardní dynamický rozsah
SH	<i>spherical harmonics</i> – sférické harmonické funkce
UUID	<i>universally unique identifier</i> – univerzální unikátní identifikátor

Úvod

Světlo je jedním z přirozených jevů, které nás obklopují každý den, aniž bychom si plně uvědomovali, jak zásadní roli hraje v našem vnímání světa. Díky světlu rozlišujeme tvary, barvy či vzdálenosti, což ovlivňuje, jak chápeme prostor a atmosféru kolem nás. Od ranního slunečního svitu s jeho jasnými chladnými tóny až po měkké hřejivé večerní světlo – každý jeho odstín ovlivňuje, jak vnímáme svět kolem nás. Stejně jako se v reálném světě světlo odráží, láme a interaguje s prostředím, v digitálních prostředích se usiluje o totéž, neboť právě osvětlení je klíčovou složkou pro vytvoření fotorealistického obrazu.

Pro získání fotorealistických výsledků se využívají metody globálního osvětlení, které zohledňují nejen světelné paprsky dopadající na objekty přímo, ale také odrazy a lomy světla, ke kterým dochází při interakci s povrchy ve scéně. Díky tomu mohou být realisticky zachyceny efekty, jako jsou stíny, nepřímé odrazy světla nebo kaustiky.

Simulace globálního osvětlení je však výpočetně náročná, zejména jde-li o dynamické scény vyžadující vykreslování v reálném čase. V praxi se proto obvykle využívá metod, které umožňují osvětlení do jisté míry předpočítat – tedy vypočítat jej pouze jednou, a to před samotným vykreslováním scény. Míra, do jaké lze osvětlení předpočítat, závisí na tom, jak dynamické jsou objekty a světelné zdroje ve scéně. Nejvíce tak z předpočítaného osvětlení těží statické objekty vystavené stálým světelným podmínkám.

Jedno z populárních řešení pro předpočítání statického osvětlení představují tzv. lightmapy – 2D textury, které uchovávají informaci o nasvícení povrchů objektů ve scéně. V závislosti na přesnosti vzorkování, resp. rozlišení textury, tak představují aproximaci globálního osvětlení. Při samotném vykreslování je pak textura namapována na příslušné povrchy a použita pro zjištění jejich osvětlení a určení výsledné barvy bez nutnosti dalších výpočetně náročných operací.

Předpočítané osvětlení tak značně zlepšuje výkon vykreslování, aniž by výrazně snižovalo vizuální kvalitu, což z něj činí nepostradatelný nástroj v interaktivních 3D prostředích jako jsou videohry či v neposlední řadě architektonické vizualizace.

Stejně myšlenka se věnuje i následující práce, která si klade za cíl prozkoumat problematiku předpočítání statického osvětlení reprezentovaného pomocí lightmap. První část práce proto nejprve uvádí čtenáře do problematiky modelování osvětlení ve 3D grafice a následně se věnuje metodám a přístupům k jeho předpočítání ve formě lightmap. Zároveň jsou představeny i možné alternativy a doplňující techniky. Specifickou je pak kapitola zabývající se metodou path tracingu, který v této práci slouží jako hlavní algoritmus pro výpočet globálního osvětlení.

Druhá část se pak soustředí na druhý cíl práce, jímž je praktické uplatnění teoretických poznatků, z nichž vychází návrh a implementace softwarového řešení pro generování lightmap. Tím je softwarový nástroj, který umožňuje automatizovaně předpočítat statické globální difúzní osvětlení pro uživatelem zadanou 3D scénu, definovanou geometrií a zdroji světla. Výstupem nástroje je pak 3D scéna, obohacená o vygenerované lightmapy se souřadnicemi UV mapování, podle kterých lze lightmapy aplikovat na geometrii.

Na závěr práce jsou pak shrnuty dosažené výsledky, diskutována známá omezení s návrhy na možná budoucí vylepšení.

1 Teoretický základ pro předpočítané osvětlení

Podstatu lidské schopnosti vnímat světlo a jeho interakci s okolními materiály bychom pravděpodobně našli hluboce zakořeněnou v našich myslích. Díky světlu jsme takřka bez přemýšlení schopni odvodit vlastnosti povrchů kolem nás – rozpoznat hrubou strukturu kamene, hebký povrch tkaniny či tekutost kapalin. Významnou roli v tomto procesu hrají stíny, které odhalují jemné detaily tvarů, dávají prostoru hloubku a pomáhají nám určit příchozí směr či intenzitu světla. Podobně intuitivní je pro člověka i chápání odražených paprsků světla. Uvedené skutečnosti lze ilustrovat na příkladu místnosti s bílými stěnami a světla, které do ní přichází skrze okno. Světlo se odráží od stěn a stropu a prosvětluje místnost, zatímco v rozích, kde těchto odrazů dopadá méně, zůstávají tmavší oblasti.

V důsledku lidského snažení o porozumění světlu jsme díky poznáním na poli fyziky dospěli k pochopení zákonitostí, které ovlivňují jeho šíření, interakci s materiály a vnímání lidským okem. Tato poznání umožnila nejen rozvoj vědeckých disciplín, ale také praktické využití v různých oblastech jako je počítačová grafika. V počítačové grafice se snažíme tyto jevy napodobovat s cílem dosáhnout fotorealistického obrazu.

1.1 Světlo v počítačové grafice

Osvětlení je základním pilířem počítačové grafiky, který zásadně ovlivňuje realističnost a dává atmosféru virtuálním scénám. Pro hlubší porozumění podstaty předpočítaného osvětlení a jeho uplatnění v praxi, je na místě čtenáře nejprve obeznámit s teoretickými koncepty pro simulaci chování světla v počítačové grafice.

1.1.1 Radiometrie

Radiometrie je odvětvím optiky, které se zabývá měřením elektromagnetického záření, kam patří i světlo. Pro přesnou reprezentaci osvětlení se tak v počítačové grafice využívá radiometrických veličin, jež umožňují světlo modelovat na základě fyzikálních principů a nacházejí tak uplatnění při řešení zobrazovací rovnice. Přestože se při simulaci osvětlení vychází z fyzikálních principů, stále se jedná pouze o aproximaci reálné fyziky.

Níže následuje výčet základních radiometrických veličin:

Zářivý tok (*radiant flux*), označovaný též jako výkon, popisuje množství zářivé energie, které projde určitou plochou za určitý časový interval. Měří se ve wattech (W). Při vykreslování se časový interval považuje za instantní, a zářivý tok tak představuje momentální výkon záření.

Intenzita záření (*irradiance*) popisuje plošnou hustotu zářivého toku v závislosti na velikosti ozařované plochy. Jednotkou je W/m^2 .

Zářivost (*radiant intensity*) označuje hustotu zářivého toku na prostorový úhel. Prostorový úhel je trojrozměrným rozšířením konceptu rovinného úhlu. Zatímco rovinný úhel měří opsanou délku souvislé množiny směrů v rovině (tj. oblouk na kružnici o poloměru 1), prostorový úhel měří opsanou plochu souvislé množiny směrů ve třírozměrném prostoru a odpovídá ploše, kterou tato množina směrů protne na povrchu koule o poloměru 1. Prostorový úhel se značí symbolem ω . Jednotkou prostorového úhlu je steradián (sr). Jednotkou zářivosti je W/sr .

Zář (radiance) popisuje hustotu zářivého toku ve směru paprsku, vztaženou k ploše kolmé na tento směr a k prostorovému úhlu. Intenzita světla dopadajícího na povrch je ovlivněna úhlem θ , pod kterým světelný paprsek dopadá na povrch. Proto je nutné provést korekci pomocí kosinového faktoru $\cos \theta$, jenž zohledňuje úbytek intenzity záření s rostoucím úhlem dopadu vzhledem k normále povrchu.

Budeme-li uvažovat prostorový úhel a plochu jako nekonečně malé, zář umožňuje popsat zářivý tok přicházející z jednoho konkrétního směru do určitého bodu na povrchu. Díky tomu můžeme jednotlivé paprsky světla uvažovat odděleně a počítat jejich dílčí příspěvek do celkového osvětlení, které dopadá do určitého bodu.

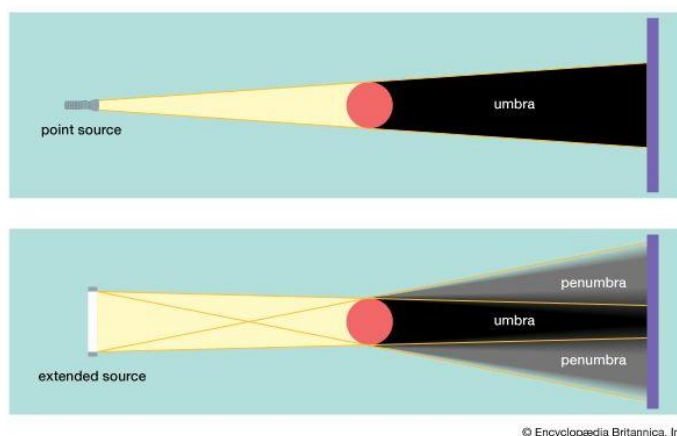
V praxi nás však zajímá celkové světlo dopadající do určitého bodu na povrchu – tedy intenzita záření. Ta je dána integrací záře ze všech směrů přichozích skrze hemisféru se středem v daném bodě, která je orientována podél normály povrchu, na němž tento bod leží.

Na stejné myšlence staví i zobrazovací rovnice (viz 1.1.4), která popisuje přenos světla ve scéně právě pomocí záře (Akenine-Möller a kol. 2018, s. 437).

1.1.2 Světelné zdroje

Pro modelování světla ve 3D grafice se využívá světelných zdrojů, z nichž každý je definován sadou parametrů určujících jeho tvar, velikost, barvu a intenzitu (Akenine-Möller a kol. 2018, s. 106-107). Modely světelných zdrojů lze rozdělit do dvou kategorií: na bezrozměrná a plošná světla (Scratchapixel nedatováno a), které se vzájemně odlišují svou vizuálně-fyzikální přesností, avšak i výpočetní náročností.

Ve skutečném světě světelné zdroje nabývají určitého rozměru, resp. tvaru – jehož povrch emituje světlo. **Bezrozměrná světla** jsou idealizovanými modely světla s nulovou plochou, které osvětlují bod na povrchu geometrie pouze z jediného směru. V anglické literatuře se někdy označují jako *delta lights* podle Diracovy delta funkce (Scratchapixel nedatováno a), která nabývá nenulové hodnoty pouze v jediném bodě. Je pro ně charakteristické, že kvůli absenci fyzického rozměru vytvářejí ostré stíny, resp. ostrý přechod mezi světlem a stínem, tzn. že osvětlovaný bod se vždy nachází buď zcela uvnitř nebo zcela vně *umbry*, zatímco přechodová oblast mezi světlem a stínem, známá jako *penumbra*, chybí úplně.



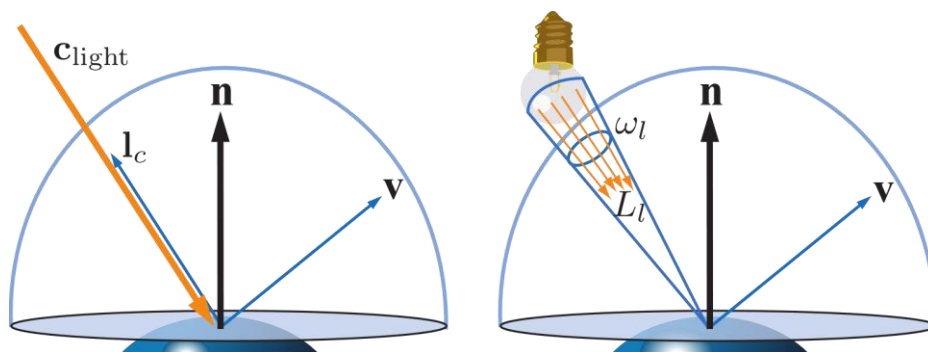
© Encyclopædia Britannica, Inc.

**Obr. 1: Stíny vytvořené bezrozměrným bodovým zdrojem světla (nahore)
a plošným zdrojem světla (dole)**

Zdroj: Britannica Kids (nedatováno)

Mezi základní typy bezrozměrných světél patří směrové světlo (*directional light*), bodové světlo (*point light* nebo též *omni light*), a reflektorové světlo (*spot light*). Směrové světlo simuluje světlo přicházející z nekonečné vzdálenosti s rovnoběžnými paprsky a jeho intenzita se v závislosti na vzdálenosti nemění. Bodové světlo vyzařuje světlo rovnoměrně všemi směry z určitého bodu ve scéně a jeho intenzita klesá s druhou mocninou vzdálenosti od zdroje. Reflektorové světlo je rozšířením bodového světla, které omezuje vyzařované světlo do kužele, jenž je definován dvěma úhly – úhlem umbry a úhlem penumbry. Úhel umbry určuje vnější hranici kužele, za kterou už světlo nepřispívá k osvětlení, zatímco úhel penumbry vymezuje vnitřní část kužele, kde je světlo na plné intenzitě. Intenzita světla směrem od penumbry k umbře postupně klesá, čímž mezi oběma kužely vzniká plynulý přechod. (Akenine-Möller a kol. 2018, s. 109–115)

Druhou kategorii představují **plošná světla**, jež jsou reprezentována tvarem emitujícím světlo ze svého povrchu, přičemž na osvětlovaný bod působí všechny paprsky, z té části plochy, která je z daného bodu viditelná. Na rozdíl od bezrozměrných světél je nyní třeba počítat osvětlení nikoliv pouze z jednoho směru, ale integrovat příspěvky světla ze všech směrů v rámci prostorového úhlu, který plošné světlo vytíná vzhledem k jednotkové hemisféře se středem v osvětlovaném bodu. V důsledku toho plošná světla přirozeně vytvářejí měkké stíny, protože intenzita osvětlení postupně klesá v závislosti na tom, jak velká část světelného zdroje je viditelná z daného bodu. (Akenine-Möller a kol. 2018, s. 377–378)

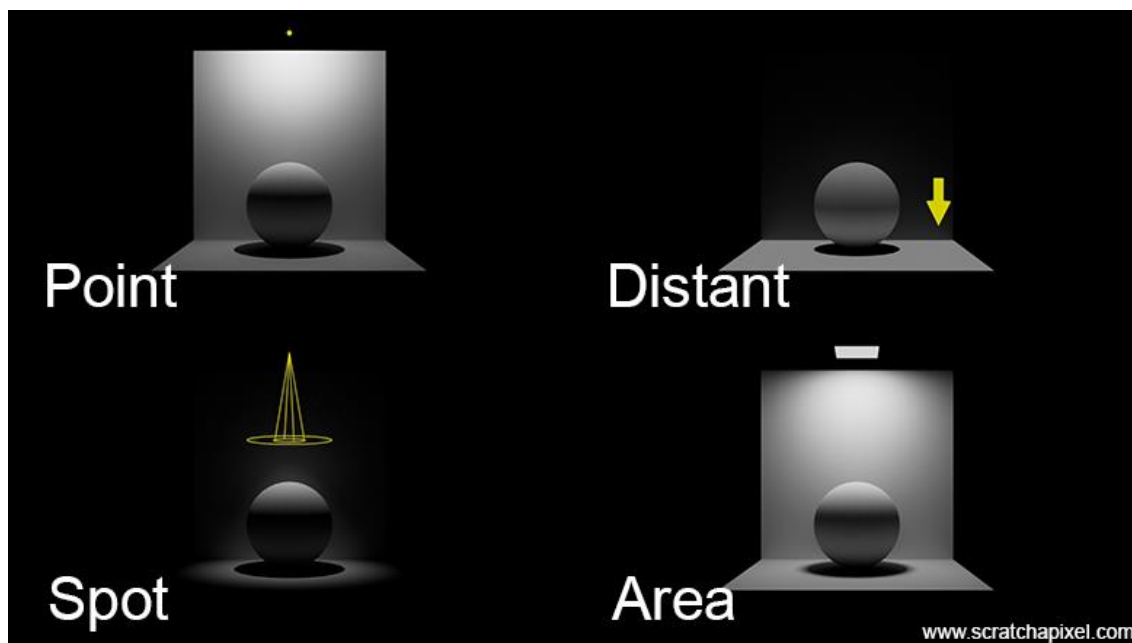


Obr. 2: Světelný zdroj osvětluje povrch, přičemž normála n určuje hemisféru všech možných směrů dopadajícího světla – vlevo jako bezrozměrný světelný zdroj, vpravo jako plošný světelný zdroj.

Zdroj: Akenine-Möller a kol. (2018, s. 378)

Jejich tvarem jsou typicky základní geometrické útvary jako obdélník, koule nebo válec, protože jejich ryze matematický popis umožňuje zjednodušit výpočet osvětlení. Mezi plošná světla patří ale i povrchy, reprezentované pomocí libovolné geometrie jako jsou trojúhelníkové sítě. (Scratchapixel nedatováno a)

Bezrozměrná světla mají ve srovnání s plošnými nižší výpočetní náročnost, protože osvětlení se počítá pouze v jediném směru, zatímco u plošných světél je nutné zohlednit jejich tvar a integrovat příspěvky světla ze všech jejich viditelných částí povrchu. Díky těmto vlastnostem byla bezrozměrná světla v minulosti hojně využívána při vykreslování v reálném čase, kdežto plošná světla sloužila spíše pokročilým simulacím, vyžadujícím fyzikální přesnost (Scratchapixel nedatováno a).



Obr. 3: Různé typy světelných zdrojů: nahoře bodový a směrový, dole reflektorový a plošný

Zdroj: Scratchapixel (nedatováno a)

S rostoucím výkonem výpočetního hardwaru a zvyšujícími se nároky na fyzikální věrnost dochází k postupnému vytlačování bezrozměrných světél ve prospěch plošných i při vykreslování v reálném čase (Akenine-Möller a kol. 2018, s. 117). Navzdory tomuto trendu však bezrozměrná světla pro svoji jednoduchost zůstávají oblíbenými a dodnes jsou používána například herními enginy (Scratchapixel nedatováno b), přičemž lze očekávat, že si své místo v praxi udrží i nadále (Akenine-Möller a kol. 2018, s. 117).

1.1.3 BRDF

Obousměrná distribuční funkce odrazu světla neboli BRDF (z anglického *bidirectional reflectance distribution function*), je matematický model, který slouží pro popis materiálových vlastností povrchu. Při metodách simulujících přenos světla dochází k výpočtům interakcí paprsků světla s povrchy ve scéně. Uvážíme-li, že paprsky světla se od povrchů odráží v závislosti na jejich materiálu, BRDF vyjadřuje vztah mezi směrem dopadajícího světla a směrem jeho odrazu. Jedná se tedy o funkci dvou proměnných – příchozího a odchozího směru, která vrací hodnotu určující, jaké množství světla je po odrazu šířeno v odchozím směru. (Boksansky 2021)

BRDF slouží jako formalismus, jenž umožňuje zapouzdřit samotnou interakci světla s materiálem jako konkrétní implementaci BRDF a kombinaci jeho parametrů. Díky tomu je možné BRDF udržovat a zkoumat nezávisle na algoritmu, který je použit pro simulaci přenosu světla. (Boksansky 2021)

Aby BRDF odpovídala fyzikálním principům, musí splňovat následující vlastnosti:

- **Helmholtzova reciprocita:** Světlo se chová stejně při prohození příchozího a odchozího směru. Jinými slovy, přichází-li paprsek z jednoho směru a odráží se do druhého, stejný odraz nastane i v opačném směru.
- **Zákon zachování energie:** Celkové množství světla odraženého nesmí překročit množství světla, které na povrch dopadlo.

V praxi se BRDF obvykle rozděluje na dvě složky, resp. dvě dílčí BRDF – *difúzní* a *spekulární*. Toto rozdělení vychází z fyzikálního chování světla při interakci s povrchem, kdy je část světla odražena přímo, což tvoří spekulární složku, zatímco zbytek světla pronikne do povrchu, kde se dále rozptyluje, a nakonec z něj vystoupí v různých směrech, čímž vytváří difúzní složku. (Boksansky 2021)

Difúzní odraz rozptyluje světlo rovnoměrně do všech směrů nad povrchem, což znamená, že jeho intenzita nezávisí na směru pohledu.

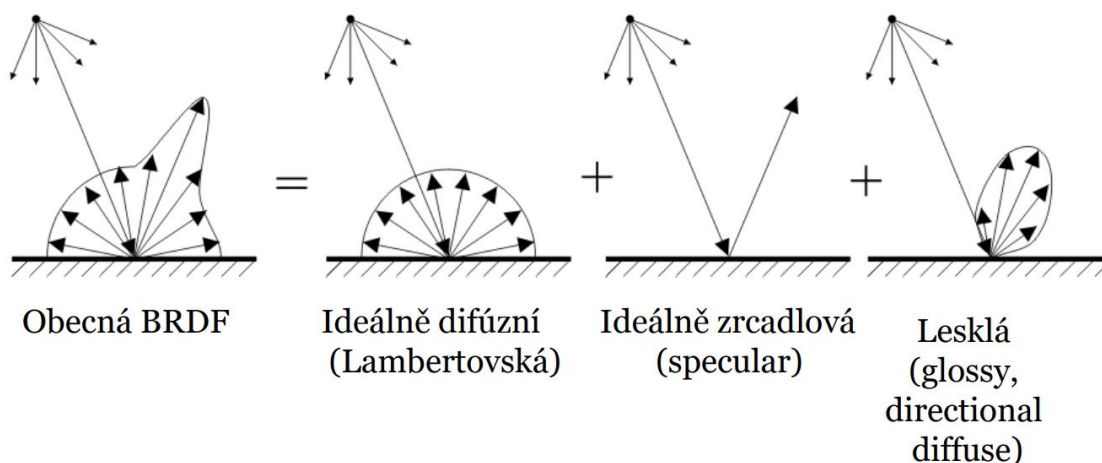
Jedním z nejjednodušších modelů BRDF je *lambertovská BRDF*, která popisuje ideálně difúzní povrch, kde je světlo rozptylováno rovnoměrně do všech směrů. Matematicky je definována jako:

$$f(l, v) = \frac{\rho}{\pi} \quad (1)$$

kde: l je směr dopadajícího světla, v je směr odraženého světla (směr pozorování) a ρ je albedo čili difúzní odrazivost udávající, jaký podíl dopadajícího světla je odražen difúzně.

Jelikož je albedo děleno konstantou π , jednou z běžných optimalizací je upravit jeho hodnotu tak, aby již zahrnovalo dělení π a k výpočtu nemuselo docházet za běhu. Alternativně lze namísto úpravy albeda předem vydělit intenzitu dopadajícího světla. (Boksansky 2021)

Spekulární odraz se řídí zákonem odrazu, kdy se světlo dopadající pod určitým úhlem odráží symetricky vzhledem k normále povrchu. Tento odraz je směrově závislý, což znamená, že jeho intenzita se mění v závislosti na úhlu pohledu. Ideálně hladké povrchy (např. zrcadla) vykazují ostrý spekulární odraz, zatímco drsnější povrchy způsobují jeho rozostření v důsledku mikroskopických nepravidlostí.



Obr. 4: Různé typy BRDF

Zdroj: Křivánek (2012)

Mimo BRDF existuje i obecnější pojem BSDF (*bidirectional scattering distribution function*), který popisuje nejen odraz, ale i přenos světla skrze materiály. BSDF se skládá ze dvou složek: BRDF pro popis odrazu a BTDF (*bidirectional transmittance distribution function*), jež popisuje, jak světlo proniká skrze průsvitné či částečně průsvitné materiály, jako je sklo, voda nebo led. BTDF však popisuje pouze to, jak se materiál chová na hranicích průniku s průsvitným materiálem, což

znamená, že stále předpokládáme, že světlo opouští povrch ve stejném bodě, kde do něj vstoupilo. (Boksansky 2021)

Ve skutečnosti se však světlo při průchodu některými materiály může šířit i pod jejich povrchem, než opět vystoupí zpět do okolního prostředí. Tento jev se nazývá podpovrchový rozptyl světla (*subsurface scattering*) a popisuje jej BSSRDF (*bidirectional subsurface scattering reflectance distribution function*) (Boksansky 2021). Podpovrchový rozptyl umožňuje realisticky simulovat materiály, jako je lidská kůže, vosk, mléko či mramor.

V praxi se termín BSDF (případně BxDF) používá jako zastřešující pojem, který zahrnuje všechny tyto modely interakce světla s materiály. (Boksansky 2021)

1.1.4 Lokální a globální osvětlení

V počítačové grafice lze modelování interakce světla s povrchy ve scéně rozdělit do dvou základních přístupů – na *lokální* a *globální osvětlení*, které se liší v rozsahu, v jakém zohledňují šíření světla ve scéně.

Lokální osvětlení zohledňuje pouze *přímé osvětlení*, tj. světlo, které dopadá na povrch přímo ze světelných zdrojů, a nebere tak v úvahu okolní prostředí – odrazy či zastínění od jiných objektů. Díky těmto omezením nemají metody lokálního osvětlení vysoké výpočetní nároky, a proto nacházejí uplatnění převážně při vykreslování v reálném čase (Akenine-Möller a kol. 2018, s. 438). Přestože takové osvětlení může být v některých případech dostačující, od metod lokálního osvětlení nelze očekávat realistické výsledky, neboť zcela ignorují světelné jevy, které vznikají vzájemným působením světla mezi objekty ve scéně.

Na druhou stranu **globální osvětlení** bere v úvahu jak *přímé*, tak *nepřímé osvětlení*, tj. světlo, které se odráží, láme a rozptyluje od ostatních objektů ve scéně (Akenine-Möller a kol. 2018, s. 438). Celý proces přenosu světla si lze představit jako opakovaný výpočet lokálního osvětlení, kdy paprsek světla po dopadu na povrch objektu nezaniká, nýbrž vytváří z povrchu sekundární zdroj světla, které se šíří prostorem dál, než doputuje ke kameře, resp. oku pozorovatele.

Výsledkem globálního osvětlení je přiblížení se reálnému chování světla, jež zahrnuje komplexní jevy, jako jsou stíny, odrazy nebo lom světla, pro jejichž výpočet je třeba brát v úvahu nejen informace o samotném objektu, který je osvětlován, nýbrž i o dalších objektech scény (Akenine-Möller et al. 2018, s. 438).

Díky této vlastnosti metody globálního osvětlení dosahují vysoce realistických výsledků. Avšak nejenže je obtížné veškeré interakce světla s povrchy objektů věrohodně replikovat, kvůli jejich složitosti se stávají i náročnými pro výpočet. V praxi se proto často hledají přijatelné kompromisy mezi fyzikální přesností a výpočetní náročností. (Akenine-Möller a kol. 2018, s. 438)

Lokální osvětlení se řídí *rovnicí odrazu*, což je speciální omezený případ kompletní *zobrazovací rovnice*:

$$L_o(p, v) = \int_{l \in \Omega} f(l, v) L_i(p, l) (n \cdot l)^+ dl \quad (2)$$

(převzato z Akenine-Möller a kol. 2018, s. 437)

kde: $L_o(\mathbf{p}, \mathbf{v})$ značí zář odcházející z povrchu v bodě $\mathbf{p}$ ve směru pozorování $\mathbf{v}$, Ω značí hemisféru všech směrů okolo bodu $\mathbf{p}$, $f(\mathbf{l}, \mathbf{v})$ značí BRDF vypočtenou pro směr pozorování $\mathbf{v}$ a aktuální přichodící směr $\mathbf{l}$, $L_i(\mathbf{p}, \mathbf{l})$ značí zář přicházející do bodu $\mathbf{p}$ ze směru $\mathbf{l}$, a $(\mathbf{n} \cdot \mathbf{l})^+$ značí skalární součin mezi směrem $\mathbf{l}$ a normálou povrchu $\mathbf{n}$, přičemž záporné hodnoty jsou nahrazeny nulou¹. (Akenine-Möller a kol. 2018, s. 437)

Z rovnice odrazu lze následně odvodit kompletní zobrazovací rovnici, kterou se řídí globální osvětlení:

$$L_o(\mathbf{p}, \mathbf{v}) = L_e(\mathbf{p}, \mathbf{v}) + \int_{\mathbf{l} \in \Omega} f(\mathbf{l}, \mathbf{v}) L_o(\mathbf{r}(\mathbf{p}, \mathbf{l}), -\mathbf{l}) (\mathbf{n} \cdot \mathbf{l})^+ d\mathbf{l}. \quad (3)$$

(převzato z Akenine-Möller a kol. 2018, s. 437)

Tímto je rovnice doplněna o člen $L_e(\mathbf{p}, \mathbf{v})$, který značí zář emitovanou povrchem v bodě $\mathbf{p}$ ve směru $\mathbf{v}$. Povrch emituje zář tehdy, je-li sám o sobě primárním zdrojem světla. Rovnice dále rozšiřuje člen $L_i(\mathbf{p}, \mathbf{l})$ do podoby:

$$L_i(\mathbf{p}, \mathbf{l}) = L_o(\mathbf{r}(\mathbf{p}, \mathbf{l}), -\mathbf{l}). \quad (4)$$

(převzato z Akenine-Möller a kol. 2018, s. 437)

Toto rozšíření znamená, že zář přicházející do bodu $\mathbf{p}$ ze směru $\mathbf{l}$ odpovídá záři odcházející z jiného bodu v opačném směru $-\mathbf{l}$. Pozice jiného bodu je pak dána funkcí $\mathbf{r}(\mathbf{p}, \mathbf{l})$, což je funkce vržení paprsku z bodu $\mathbf{p}$ ve směru $\mathbf{l}$. (Akenine-Möller a kol. 2018, s. 437)

Z definice zobrazovací rovnice tedy vyplývá, že pro určení barvy povrchu v bodě $\mathbf{p}$ je potřeba určit zář, která z bodu odchází ve směru pozorování $\mathbf{v}$. Zář přicházející do jednoho bodu však musí odcházet z nějakého jiného bodu, což z rovnice činí nekonečně rekurzivní proces. (Akenine-Möller a kol. 2018, s. 438)

Aby však bylo možné simulaci globálního osvětlení realizovat, v praxi je tato rekurze omezena na konečný počet iterací. K aproximaci tohoto procesu se běžně využívají *metody Monte Carlo*, které umožňují stochastické vzorkování světelných cest a tím numericky řešit integrální formu zobrazovací rovnice.

1.2 Předpočítané globální osvětlení

Globální osvětlení nabízí vysoce kvalitní fotorealistické výsledky. Jeho výpočetní metody jsou však výpočetně náročné, a proto nevhodné pro vykreslování v reálném čase, při kterém hraje klíčovou roli rychlá odezva. Pro dosažení plynulého nepřerušovaného zážitku, by se prodleva mezi snímky měla pohybovat v rozmezí 16-30 ms, přičemž v tomto čase je zároveň třeba obstarat i ostatní systémy, které simulují virtuální prostředí, jako je v případě herních enginů např. zpracování uživatelského vstupu, fyzikální systém či v neposlední řadě řízení umělé inteligence (Seyb-Möller a kol. 2020, s. 1).

Z pohledu výpočetní náročnosti, resp. dlouhé výpočetní doby, je globální osvětlení přijatelné například u počítačově generovaných filmů, kde lze všechny snímky jednou předem plně

¹ Člen $(\mathbf{n} \cdot \mathbf{l})$ odpovídá kosinovému faktoru $\cos \theta$, který určuje intenzitu osvětlení povrchu v závislosti na úhlu, pod kterým světlo dopadá. Nejvyšší intenzita je dosažena při kolmém dopadu světla, zatímco při větších úhlech se intenzita úměrně zeslabuje.

vykreslit a poté opakovaně přehrávat bez zpoždění, neboť obsah jednotlivých snímků zůstává při dalších přehráváních stejný. Výzvou však zůstávají interaktivní aplikace, jako jsou videohry či architektonické vizualizace, které nabízejí jistou míru interaktivity, jež obvykle spočívá minimálně ve volném pohybu kamery. To znamená, že při každém pohybu kamery či jiné změně ve scéně by muselo dojít k opakovanému výpočtu osvětlení. Tento proces by byl nejen výpočetně náročný a doprovázený vysokým zpožděním, ale v případě statické povahy osvětlení scény také zbytečně neefektivní.

Aby tak bylo možné využívat efekty globálního osvětlení i zde, ubírá se většina moderních herních enginů k řešením, která umožňují informace o osvětlení zcela nebo částečně předpočítat ještě před samotným zobrazováním (Seyb-Möller a kol. 2020, s. 1). Tím se většina výpočetní zátěže přesouvá do fáze produkce, zatímco za běhu aplikace se předpočítané informace pouze načítají a využívají k nasvícení povrchů scény. Proces předpočítání osvětlení se obvykle označuje jako „baking“ (Akenine-Möller a kol. 2018, s. 473).

Navzdory neustále rostoucímu výkonu hardwaru, pokroku v oblasti grafických algoritmů a snaze o výpočet globálního osvětlení v reálném čase, výkon průměrných uživatelských zařízení stále nedosahuje potřebné úrovně, a stejně tak i v budoucnu lze očekávat, že svou efektivitou stěží překoná předpočítaná řešení (Seyb-Möller a kol. 2020, s. 1).

Podstata předpočítání osvětlení spočívá v úvaze, že zobrazované scény, byť interaktivní, jsou z větší části statické. To znamená, že lze definovat stavy, při kterých zůstává velké množství geometrie a povrchových materiálů v neměnném stavu, přičemž jsou vystaveny stálým světelným podmínkám. V tomto stavu je pak možné osvětlení do jisté míry předpočítat. Jakmile však dojde k sebemenší změně ve statické části scény, je třeba výpočet osvětlení provést znovu, jelikož původní předpočítaná data již nebudou platná.

V praxi to znamená, že objekty ve scéně jsou rozděleny na statické a dynamické. Jako příklad statických objektů lze uvést např. terén, stěny či střechy, jejichž poloha ve scéně se nemění, zatímco pod dynamickými objekty si lze představit např. postavu, vozidlo či jiné předměty, které se mohou volně pohybovat. Při předpočítání osvětlení se pak v úvahu berou pouze statické objekty, jejichž povrchy mohou bránit průchodu světla, jakož i mohou emitovat či odrážet paprsky nepřímého osvětlení. Naopak dynamické objekty osvětlení pouze přijímají, což se děje na základě předpočítaných informací. V některých případech – zejména u menších objektů – lze absenci dynamických objektů při výpočtu osvětlení zcela zanedbat, anebo jejich vliv na stínování za běhu aproximovat komplementárními dynamickými metodami, které napodobují efekty globálního osvětlení, jako jsou *shadow mapping* či *ambient occlusion*. (Akenine-Möller a kol. 2018, s. 473–474)

Nežli přistoupíme k samotným metodám předpočítaného osvětlení, je vhodné zmínit, že přístupy k jeho řešení lze na základě odrazivosti povrchů rozdělit do dvou základních kategorií: na difúzní a spekulární osvětlení. Toto dělení vychází z povahy posledního odrazu paprsku před jeho dopadem do kamery.

Metody určené pro difúzní osvětlení předpokládají, že světlo dopadající na povrch se v rámci hemisféry nad povrchem mění plynule s nízkou frekvencí. Naproti tomu u spekulárního osvětlení dochází v závislosti na směru k výrazně rychlejším změnám dopadajícího světla. V případě spekulárního osvětlení navíc světlo přichází pouze z omezeného prostorového úhlu, což ve srovnání s difúzními materiály klade výrazně odlišné požadavky na reprezentaci osvětlení.

Vzhledem k jejich odlišné povaze je proto výhodné oba typy osvětlení v rámci metod předpočítaného osvětlení řešit individuálně. (Akenine-Möller a kol. 2018, s. 472–743, 497–498)

Řešení předpočítaného spekulárního osvětlení na rozdíl od difúzního vyžaduje sofistikovanější přístup. Z tohoto důvodu je spekulární osvětlení v práci zmiňováno pouze okrajově, zatímco hlavní pozornost je věnována difúznímu osvětlení, kdy je u všech povrchů předpokládána lambertovská BRDF.

V rámci praktického využití předpočítaného osvětlení je žádoucí, aby výsledná data byla uložena ve formátech, které umožňují rychlý a efektivní přístup. Slovy Yuriye O'Donnella (2018, s. 39), předpočítané osvětlení představuje cache, jež uchovává částečné řešení zobrazovací rovnice, jež je indexováno pozicí osvětlovaného bodu ve scéně. Z pohledu ukládaných dat je předpočítané osvětlení velice flexibilní, neboť mu vychází vstříc samotná zobrazovací rovnice (O'Donnell 2018, s. 39). Z její definice plyne, že výsledná zář je dána skládáním příspěvků z různých světelných zdrojů a interakcí světla ve scéně, což v praxi umožňuje kombinovat různá dílčí řešení, jež mohou být vypočítána jak offline, tak v reálném čase (O'Donnell 2018, s. 39).

1.3 Lightmapping

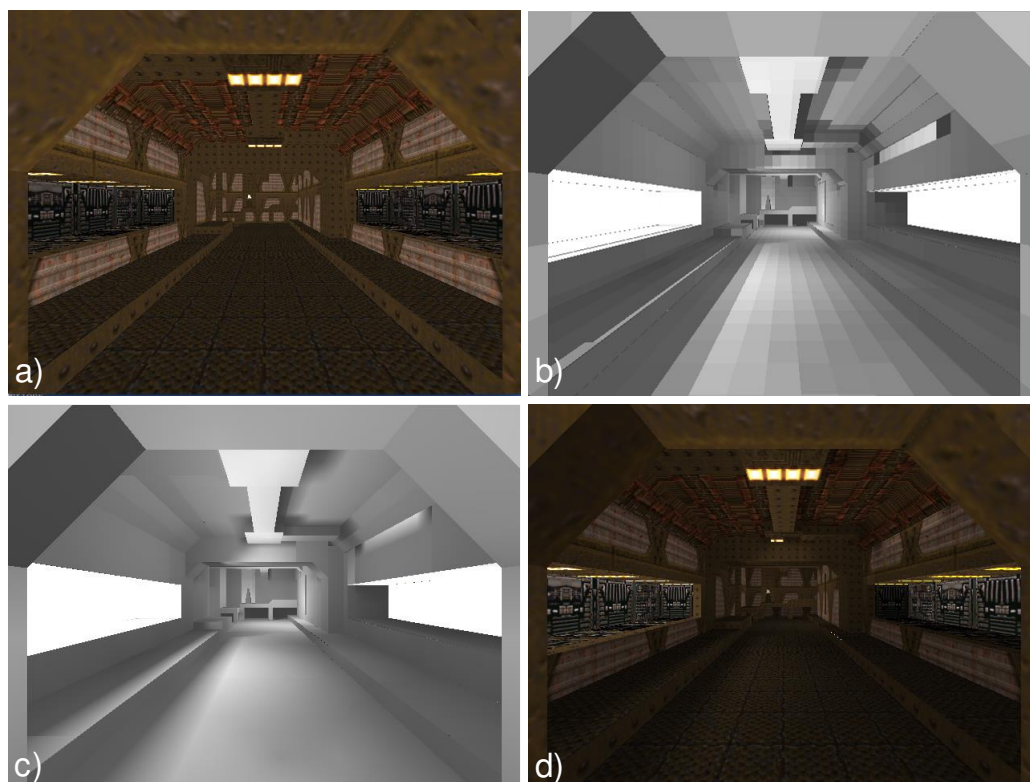
Následující kapitola se věnuje popisu různých datových struktur, jež lze využít pro uložení předpočítaného osvětlení, přičemž největší pozornost je věnována *lightmappingu*, jakožto stěžejnímu konceptu v kontextu této práce.

Lightmapping představuje metodu, která slouží pro ukládání a následné zobrazování předpočítaného osvětlení na povrchu geometrie za pomoci 2D textur – tzv. *lightmap* (Akenine-Möller a kol. 2018, s. 484). V češtině by šlo lightmapping volně přeložit jako „mapování světla“, protože tato metoda spočívá v tom, že se světelné informace scény uloží do textur. Každý *texel* (*texture element* – analogie pixelu rastrového obrazu) pak v případě lightmapy nese údaje o osvětlení konkrétní části povrchu.

Na rozdíl od jiných textur, jako jsou například textury albeda, které se na površích geometrie mohou opakovat, přičemž sdílejí stejnou oblast textury, je při práci s osvětlením nutné vycházet z předpokladu, že osvětlení může být pro každý povrch jedinečné (Akenine-Möller a kol. 2018, s. 485). Proto je nedílnou součástí lightmap také *unikátní UV parametrizace*, která zajišťuje tvorbu takového UV mapování², kde každý povrch má po rozvinutí do 2D prostoru na lightmapě vyhrazeno své vlastní místo, jež se nepřekrývá s žádným jiným povrchem (Akenine-Möller a kol. 2018, s. 485). Tím je každému povrchu přidělena konkrétní oblast texelů na lightmapě, kam lze uložit předpočítané osvětlení.

Pomocí UV mapování lze osvětlení aplikovat na příslušné povrchy. Za předpokladu, že geometrie je reprezentována pomocí trojúhelníkových sítí, které představují základní a nejběžnější formu reprezentace pro GPU, každý vrchol trojúhelníku nese souřadnici UV mapování, jejichž interpolací uvnitř trojúhelníku lze pro libovolný bod na jeho povrchu vzorkovat předpočítané osvětlení z lightmapy.

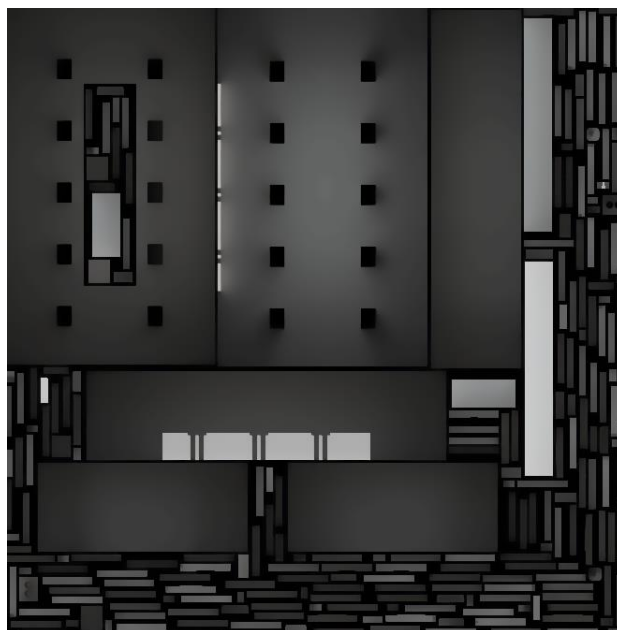
² UV mapování je proces přiřazení souřadnic z dvourozměrného texturového prostoru (označeného jako U a V) jednotlivým bodům geometrie (trojúhelníkové sítě), aby bylo možné texturu správně promítnout na její povrch.



Obr. 5: Použití lightmap ve videohře Quake (1996) od společnosti id Software: (a) albedo, (b) lightmapa s nearest-neighbor filtrováním, (c) lightmapa s bilineárním filtrováním, (d) kombinace albeda a bilineárně filtrované lightmapy

Zdroj: Bush (2015)

Je důležité zdůraznit, že osvětlení je spojitou veličinou, která se plynule mění po povrchu geometrie. Ukládáním předpočítaného osvětlení do textur však dochází k jeho diskretizaci. Výsledná rekonstrukce osvětlení je proto pouze aproximací skutečné hodnoty, přičemž její přesnost závisí na hustotě texelů vůči velikosti povrchu, který daná část textury pokrývá.



Obr. 6: Příklad textury lightmapy zobrazené v texturovém prostoru

Zdroj: vlastní zpracování

Při volbě rozměru lightmapy je zásadní zohlednit velikost povrchů, jež lightmapa pokrývá, neboť rozměr určuje celkovou plochu, do které lze povrchy rozprostřít, a zároveň maximální frekvenci, při jaké lze osvětlení reprezentovat. Povrchy se na lightmapě rozprostírají v určitém měřítku, které určuje, kolik texelů připadne na jednotku plochy daného povrchu. To znamená, že volba nižšího měřítka vede k vyšší kvalitě osvětlení, protože na povrch připadne více texelů. Avšak i při nízkém měřítku nemusí být kvalita osvětlení dostatečná, pokud je rozměr lightmapy příliš malý. V takovém případě sice povrch zabere větší část z celkové plochy lightmapy, ale jednotlivé texely mohou být příliš velké na to, aby zachytily osvětlení s dostatečnou přesností. Pro zachování dostatečné kvality je proto u větších ploch vhodné zvolit nejen nižší měřítko, ale i větší rozměr lightmapy.

Dalším faktorem je také charakter osvětlení, jež se do textur ukládá. Nepřímé osvětlení, které je díky své typicky nízké frekvenci hladké, nevyžaduje vysoké rozlišení. Naopak u přímého osvětlení, zejména v případech ostrých stínů, je vyšší rozlišení nutné, aby bylo možné zachytit detaily s dostatečnou přesností. Podobně jako u jiných textur, například textur albeda, hraje důležitou roli i vizuální důležitost povrchu, jež je ovlivněna například tím, jak blízko se k němu může přiblížit kamera. U vzdálených či méně viditelných ploch si předpočítané osvětlení obvykle vystačí s nižším rozlišením. Volba vhodného rozlišení lightmapy je tedy výsledkem kombinace estetických a technických požadavků, kdy cílem je optimalizovat spotřebu paměti a výpočetní náročnost spojenou s předpočítáním osvětlení, a přitom zachovat dostatečnou úroveň detailů, aby bylo osvětlení vizuálně přesvědčivé.

Vlivem diskretizace může docházet ke vzniku viditelných ostrých přechodů, zejména pokud je rozlišení lightmapy nízké. V praxi se proto obvykle pro lightmapy využívá *bilinéárního filtrování* (Akenine-Möller a kol. 2018, s. 485), které při vykreslování textury interpoluje hodnoty ze 4 sousedních texelů a vytváří tak hladší přechody. Bilineární filtrování navíc těží z přímé podpory hardwarové akcelerace na straně GPU. Pro dosažení ještě jemnějších přechodů se využívá *bikubického filtrování* (Rahteenko 2018), které zohledňuje hodnoty z 16 sousedních texelů a interpoluje je pomocí kubických funkcí, avšak za cenu o něco vyšší výpočetní náročnosti.



Obr. 7: Porovnání povrchu osvětleného stejnou lightmapou za využití bilinéárního filtrování (vlevo) a bikubického filtrování (vpravo)

Zdroj: Rahteenko (2018)

Prozatím byla zmíněna problematika diskretizace a filtrování lightmap, avšak doposud nebylo zmíněno, jakým způsobem vzniká unikátní UV parametrizace povrchu geometrie. Při jejím návrhu je nutné oba tyto aspekty brát v úvahu, protože kladou specifické požadavky na její rozložení v texturovém prostoru.

Tento proces spočívá v rozdělení povrchu geometrie na menší segmenty zvané „charts“. Ty mohou být vytvořeny buď pomocí heuristických algoritmů nebo manuálně v modelovacím softwaru. Používá-li geometrie i jiných textur, obvykle se vychází z již existujícího UV mapování, které určuje rozdělení povrchu geometrie³. Každý chart je následně parametrizován samostatně tak, aby se jeho části nepřekrývaly v texturovém prostoru. (Akenine-Möller a kol. 2018, s. 485)

Po dokončení parametrizace jednotlivých chartů následuje jejich rozmístění v rámci texturového *atlasu* – společné textury, která ukládá více menších textur, v tomto případě chartů. V kontextu lightmap budeme tento atlas označovat jako *lightmapový atlas*.

V této fázi je nutné zajistit nejen, aby se charty navzájem nepřekrývaly, ale aby mezi nimi byla dostatečná mezera (*padding/gutter*), která umožní výplň texelů na okrajích chartu, čímž se zajistí jejich správné vzorkování při filtrování (Akenine-Möller a kol. 2018, s. 485). Velikost mezery musí odpovídat oblasti vzorkování dané použitým filtrem – mřížka 2×2 pro bilineární filtrování, které vzorkuje ±1 sousední texel v každém směru nebo mřížka 4×4 pro bikubické filtrování, které vzorkuje ±2 sousední texely v každém směru. V opačném případě by mohlo docházet k úniku světla ze sousedních chartů nebo z nevyužitého texturového prostoru.

Optimální rozmístění chartů do lightmapového atlasu je NP-úplným problémem, pro který neexistuje žádný algoritmus, jenž by vedl na ideální řešení s polynomiální složitostí. V praxi se proto využívají řešení s precizně vyladěnými heuristikami a pečlivě optimalizovaným kódem, díky nimž lze generovat rozmístění chartů rychle a s dostatečnou kvalitou. (Akenine-Möller a kol. 2018, s. 486)

Dalším problémem souvisejícím s parametrizací geometrie pro lightmapping představují švy (*seams*) – tedy hrany sdílené mezi dvěma charty. Každý chart je však parametrizován samostatně, takže tyto hrany zůstávají spojené pouze ve 3D prostoru, zatímco v UV prostoru jsou oddělené. V důsledku toho nemusí být předpočítané osvětlení na obou stranách stejné, což vede k vizuálním artefaktům v podobě nespojitostí, které se projevují jako ostré přechody v osvětlení povrchu. (Akenine-Möller a kol. 2018, s. 486)

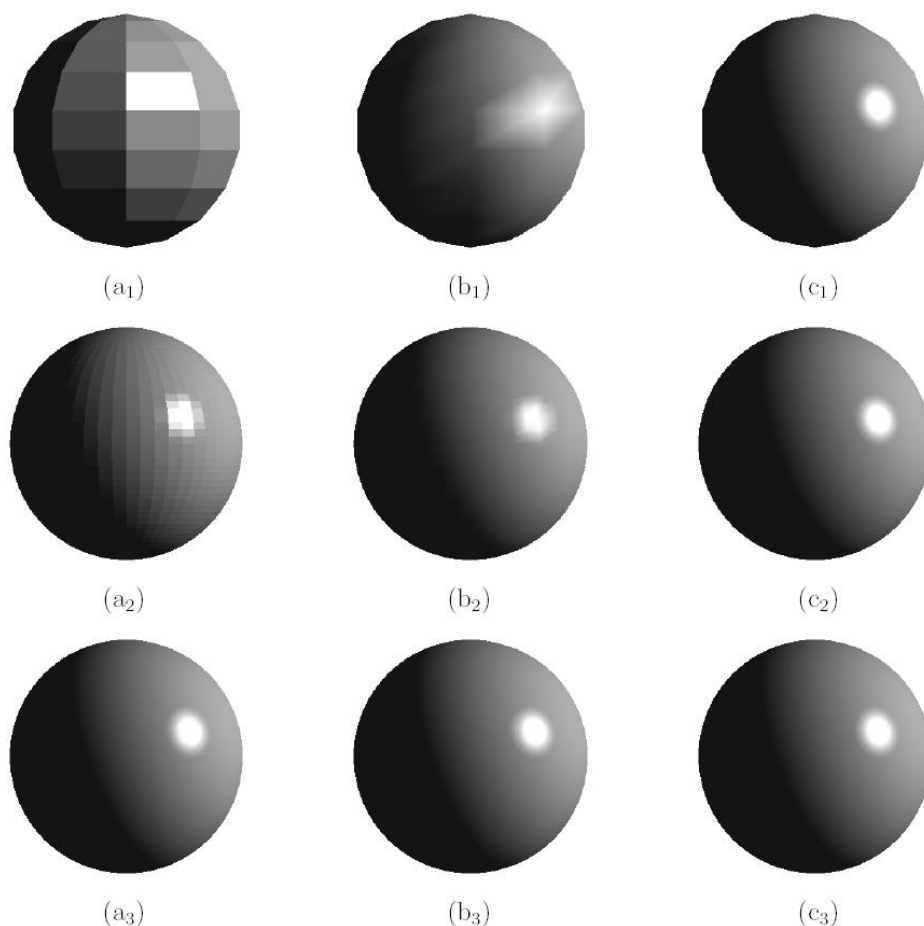
Tento problém lze částečně řešit například umístěním švů do méně viditelných oblastí, to však vyžaduje manuální zásah návrháře geometrie. (Akenine-Möller a kol. 2018, s. 486)

V praxi se proto používají různé přístupy, jež umožňují tento problém řešit automatizovaně. Iwanicki (2013, s. 8–12) popisuje řešení, které po vypočítání hodnot v lightmapě provádí postprocessing, jenž minimalizuje rozdíly v interpolovaných hodnotách na obou stranách švu pomocí metody nejmenších čtverců.

³ Při UV mapování může docházet k „roztržení“ dvou částí povrchu v UV prostoru, zatímco ve 3D prostoru zůstávají spojeny. Vyžadují-li dva sousední trojúhelníky odlišné UV souřadnice pro tentýž vrchol, musí být tento vrchol duplikován, aby každý trojúhelník měl svou vlastní instanci s odpovídajícími UV souřadnicemi.

1.3.1 Alternativní a rozšiřující metody

Příbuznou alternativou k lightmapám je *vertex baking*, čili uložení předpočítaného osvětlení přímo do vrcholů jakožto jeden z jeho atributů (Schäfer 2012, s. 193). Při vzorkování osvětlení na povrchu objektu se hodnoty uložené ve vrcholech trojúhelníku interpolují, což umožňuje určit přibližnou hodnotu osvětlení pro jakýkoliv bod ležící uvnitř trojúhelníku. Výsledné osvětlení svým charakterem připomíná *Gouraudovo stínování*⁴ a trpí stejnými vizuálními nedostatky.



Obr. 8: Různé úrovně, na kterých je osvětlení počítáno: (a) ploché stínování: jednou pro celý polygon, (b) Gouraudovo stínování: pro každý vrchol s interpolací mezi nimi, (c) Phongovo stínování: pro každý pixel výsledného obrazu

Zdroj: Xiaoyao Wen Gongzuo Shi (2011)

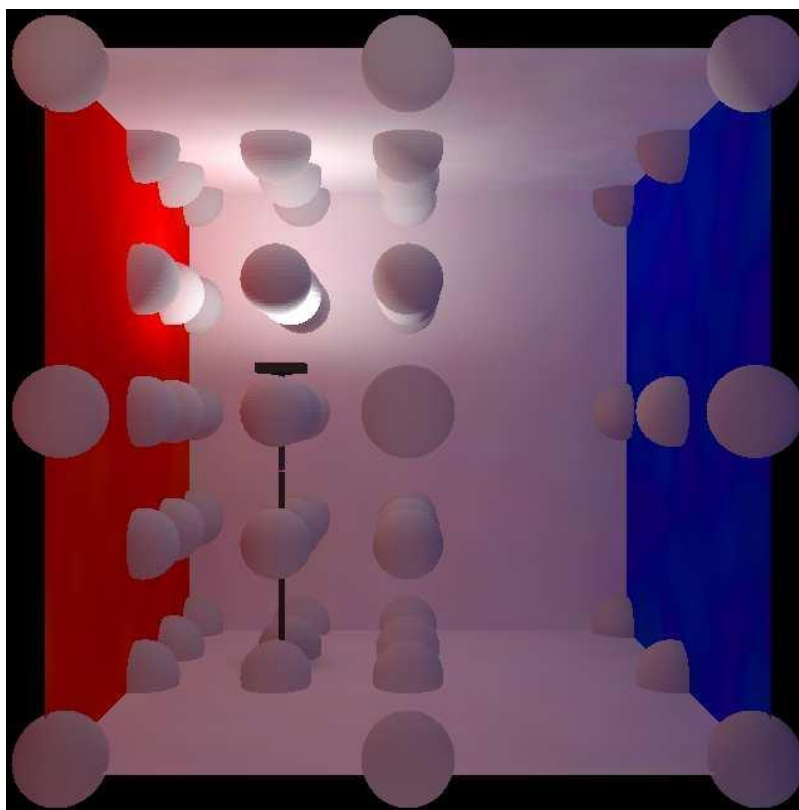
Výhodou tohoto přístupu je, že nevyžaduje další textury ani UV parametrizaci. Avšak přestože je tento způsob jednodušší na implementaci a vyžaduje méně paměti než lightmapy, nese s sebou svá omezení, jež jej činí nepraktickým. Opět dochází k diskretizaci osvětlení, která je v tomto případě ovlivněna hustotou vrcholů v síti. Na detailních, ev. rovnoměrně teselovaných sítích může poskytovat dostatečné kvalitní výsledky (Akenine-Möller a kol. 2018, s. 487), kdežto na řídkých sítích může vést k viditelným artefaktům, spočívajícím v nerovnoměrném osvětlení či

⁴ Gouraudovo stínování (Gouraud 1998) se na konci 90. let 20. století hojně využívalo ve videohrách pro dynamický výpočet osvětlení (příspěvatelé Giant Bomb nedatováno). Díky interpolaci osvětlení na úrovni vrcholů nabízelo nižší výpočetní náročnost, což jej činilo vhodné pro omezený výkon tehdejšího hardwaru. V průběhu 21. století však bylo postupně nahrazeno přesnějším Phongovým stínováním (Phong 1975), jež provádí výpočty na úrovni pixelů.

nedostatečném zachycení jemných detailů, například stínů. Vysoce detailní geometrie navíc zvyšuje nároky na paměť sama o sobě a současně vyžaduje vyšší výpočetní výkon pro její zpracování (Akenine-Möller a kol. 2018, s. 487). Z výše uvedených důvodů není toto řešení v dnešní praxi příliš populární (Akenine-Möller a kol. 2018, s. 487).

Metody, které byly prozatím představeny umožňují ukládat předpočítané osvětlení na úrovni povrchů, v praxi se však využívají i metody, jež předpočítávají osvětlení objemově na úrovni 3D prostoru. To znamená, že osvětlení lze vzorkovat v libovolném bodě scény, což umožňuje osvětlovat i dynamické objekty, které při předpočítání osvětlení nebyly brány v úvahu.

Jednou těchto metod je tzv. *irradiance volume*, představená Gregerem a kol. (1998), jež funguje na principu mřížky, jež ohraničuje osvětlovaný prostor. V každém bodu mřížky se nachází tzv. *light probe* neboli světelná sonda, která ukládá informace o nepřímém osvětlení, jež do daného bodu přichází ze všech okolních směrů. Chceme-li osvětlovat libovolný bod v rámci prostoru, je výsledné osvětlení vypočteno interpolací osvětlení z nejbližších bodů mřížky.



Obr. 9: Irradiance volume s pravidelně rozmístěnými light probes, znázorněnými pomocí koulí.

Zdroj: Greger a kol. (1998, s. 9)

Původní metoda využívá pravidelné dvouúrovňové adaptivní mřížky, která osvětlení interpolovala na základě trilineární interpolace (Greger a kol. 1998, s. 8–10). Mřížku však lze reprezentovat i pomocí jiných datových struktur dělících prostor, jako je např. *octree* (Cupisz 2012, s. 4).

Hlavní nevýhodou prostorových metod je vysoká spotřeba paměti. Zatímco u lightmap roste se čtvercem jejich rozlišení, u pravidelných prostorových mřížek roste s krychlí. Proto se pro ně obvykle volí nižší rozlišení, které však vede k méně přesným výsledkům. Tento problém částečně řeší adaptivní mřížky, jež zvyšují rozlišení pouze tam, kde je to nezbytné, a šetří paměť

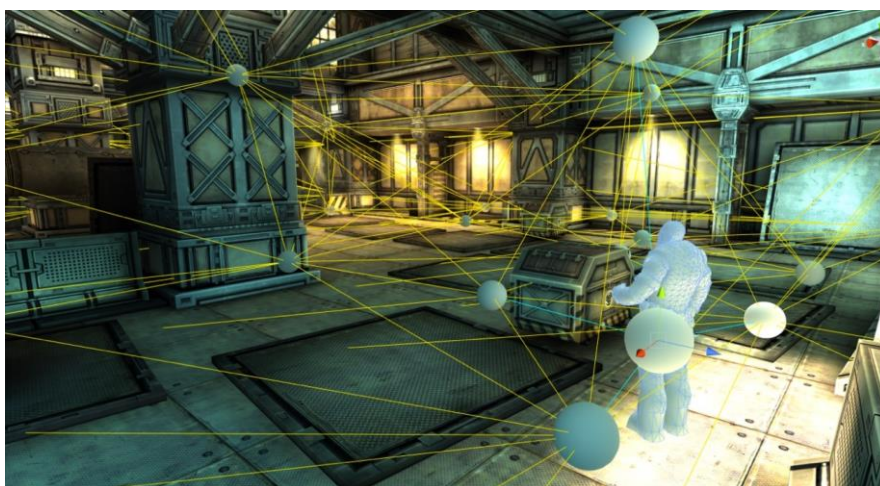
v oblastech s nižší frekvencí osvětlení. Adaptivní přístup však činí vzorkování osvětlení výpočetně náročnější. (Akenine-Möller a kol. 2018, s. 488)

Další problém pravidelných mřížek spočívá v tom, že vlivem pravidelnosti se některé z jejích bodů musejí ocitnout uvnitř neprostupných překážek. To pak u dynamických objektů může způsobit, že v blízkosti překážky tmavnou, protože vzorkují světlo ze zastíněné light probe. Podobný problém nastává, když se dynamický objekt pohybuje na rozhraní dvou zcela odlišně osvětlených prostředí, které jsou odděleny stěnou. (Cupisz 2012, s. 6)

Mezi běžná řešení tohoto problému patří posun vzorkovaného osvětlovaného bodu podél jeho povrchové normály nebo úpravou vah, jež ovlivňují, jak která light probe přispívá k osvětlení v daném bodě, resp. snížením jejich vlivu, nalézají-li se uvnitř překážek (Akenine-Möller a kol. 2018, s. 489). Pokročilejší řešení popisuje Hooker (2016, s. 30–49), kdy je oblast relevantních light probes ohraničena pomocí rovin tvořících konvexní mnohostěn, který zabraňuje vzorkování z light probes nacházejících se mimo něj.

Mřížka nemusí být pravidelná – mezi populární řešení patří i použití nepravidelně rozmístěných bodů, jež se propojí pomocí *Delaunayho tetradrizace*. Tím vznikne síť složená z čtyřstěnů, uvnitř kterých lze interpolovat osvětlení. Oproti pravidelné mřížce je však obtížnější nalézt čtyřstěn, ve kterém se nachází osvětlovaný bod. Postup začíná zvolením výchozího čtyřstěnu, pro který se spočítají barycentrické souřadnice bodu. Na základě nich lze určit, zda bod leží uvnitř daného čtyřstěnu, nebo jestli je třeba pokračovat hledáním v některém z jeho sousedů. Tento proces se opakuje, dokud není nalezen čtyřstěn, ve kterém se osvětlovaný bod nachází. Při hledání hrají klíčovou roli barycentrické souřadnice – nejenže určují polohu bodu vůči vrcholům čtyřstěnu, ale také pomáhají rozhodnout, kterým směrem se při hledání posunout dál, a ve výsledku slouží i jako váhy, na jejichž základě je osvětlení interpolováno mezi vrcholy čtyřstěnu. (Cupisz 2012)

Vyhledávání pak lze akcelarovat např. pomocí uchovávání čtyřstěnu, který bodu odpovídal při vykreslování předchozího snímku (Cupisz 2012, s. 18), volbou čtyřstěnu ve středu sítě jako výchozího (Cupisz 2012, s. 29), nebo pomocí dodatečné prostorové datové struktury, která při hledání pomůže zvolit vhodný výchozí čtyřstěn (Akenine-Möller a kol. 2018, s. 489).



Obr. 10: Nepravidelně rozmístěné light probes (znázorněné jako koule) propojené do sítě tvořené čtyřstěny pomocí tetradrizace – snímek z prostředí herního engine Unity zobrazující scénu videohry Shadowgun od společnosti MADFINGER Games

Zdroj: Cupisz (2012, s. 15)

1.4 Veličiny a formáty předpočítaného osvětlení

V této kapitole jsou představeny radiometrické veličiny a různé formáty, které se používají k ukládání hodnot předpočítaného osvětlení do nejmenších adresovatelných jednotek v odpovídající datové struktuře. V případě lightmapy je nejmenší adresovatelnou jednotkou texel.

Základní myšlenkou lightmappingu je vyjmout z výpočtu osvětlení ty složky, které jsou výpočetně náročné, ale zároveň statické – tedy nemění se v čase. To znamená, že během dochází pouze ke čtení dříve vypočítané hodnoty, která je úplným nebo částečným řešením poslední interakce světla před jeho dopadem na povrch. To odpovídá výpočtu rovnice odrazu, jež integruje zář $L_i(p, l)$ ze všech směrů přichozích skrze hemisféru nad osvětlovaným bodem, která je následně modulována BRDF materiálu $f(l, v)$ a kosinovým faktorem $(n \cdot l)^+$.

V minulosti byly motivací pro lightmapping především stíny a přímé osvětlení (Väänänen 2024), které tehdy pro hardware představovaly značnou zátěž. To dokazuje například implementace generování lightmap v herním enginu *Insanity3D* (Bačík nedatováno), který byl použit ve videohře *Hidden & Dangerous* z roku 1999 vyvinuté společností Illusion Softworks. Díky jeho pozdějšímu zpřístupnění jako open-source lze dnes studovat způsob předpočítání osvětlení v komerčních titulech té doby.

S technologickým pokrokem a zvyšujícím se výkonem GPU se však situace změnila. V současnosti je hlavním důvodem použití lightmap především náročnost výpočtu nepřímého osvětlení, zatímco přímé osvětlení včetně stínů může být počítáno dynamicky v reálném čase (O'Donnell 2018, s. 9).

Není to však pravidlem. Rozhodnutí, zda předpočítat i přímé osvětlení, závisí na povaze světelných zdrojů (zda jsou dynamické, nebo statické) a rovněž na kompromisu mezi výpočetní a paměťovou náročností. Zatímco přímé osvětlení se stíny vyžadují pro dosažení přijatelné kvality vyšší rozlišení, u nepřímého osvětlení si lze vystačit s výrazně nižším rozlišením bez výrazného dopadu na vizuální dojem.

V kontextu rekonstrukce osvětlení z lightmapy je irelevantní, zda přímé osvětlení je či není jeho součástí. Pokud není, je třeba jej vypočítat dynamicky za běhu a jeho příspěvek sečíst společně s nepřímým osvětlením z lightmapy. Dynamický výpočet sice zvyšuje výpočetní náročnost, ale na druhou stranu eliminuje degradaci osvětlení, která by jinak vznikla diskretizací osvětlení do lightmapy.

Hodnoty uložené v lightmapě by měly splňovat dva základní požadavky: zachovat dostatečnou přesnost a umožnit rychlou rekonstrukci. Dostatečnou přesnost pak lze posoudit například na základě vizuální kvality, která je dostatečná pro konkrétní potřeby aplikace.

1.4.1 Základní „ploché“ lightmapy

Návrh konkrétních hodnot, jež se do lightmapy ukládají, vychází z rovnice odrazu. Klíčové je rozlišit, které členy jsou v době výpočtu lightmapy neznámé a které jsou naopak předem dané, zůstávají při stínování bodu konstantní a lze je tedy předpočítat. Je-li výsledná barva bodu určena integrálem přes všechny směry přichozího světla skrze hemisféru kolem daného bodu, kde se všechny členy uvnitř integrálu násobí, lze konstantní členy vytknout a umístit je před samotný

integrál. Navíc díky této vlastnosti lze více konstantních členů uložit kompaktně do jediné hodnoty jako jejich součin.

Zaměříme-li se na konkrétní podobu rovnice odrazu pro lambertovskou BRDF, dostáváme:

$$L_o(p, v) = \int_{l \in \Omega} \frac{\rho}{\pi} L_i(p, l) (n \cdot l)^+ dl \quad (5)$$

$$= \frac{\rho}{\pi} \int_{l \in \Omega} L_i(p, l) (n \cdot l)^+ dl \quad (6)$$

Po dosazení je patrné, že z rovnice zcela zmizela neznámá v podobě směru pozorování. Vytknutím faktoru $\frac{\rho}{\pi}$ před integrál odpovídá současný integrál výpočtu intenzity záření E v bodě p .

$$E(p) = \int_{l \in \Omega} L_i(p, l) (n \cdot l)^+ dl \quad (7)$$

Navíc za předpokladu, že je známa i povrchová normála osvětlovaného bodu, závisí už výpočet intenzity záření pouze na směrové distribuci záře přichází skrze hemisféru. V takovém případě jsou všechny trojúhelníky povrchu považovány za lokálně ploché, tedy že normála v rámci každého trojúhelníku zůstává buď konstantní, nebo se plynule mění s nízkou frekvencí prostřednictvím lineární interpolace normál definovaných ve vrcholech trojúhelníku.

Avšak je-li základním předpokladem předpočítaného osvětlení, že osvětlení scény zůstává statické, je zcela samozřejmé, že směrová distribuce přichází záře je rovněž předem známa.

Z následujícího odvození je patrné, že v případě povrchů s lambertovskou BRDF lze integrál intenzity záření předpočítat. Výsledná barva osvětlovaného bodu je následně dána prostým součinem intenzity záření a barvy povrchu, resp. jeho albeda.

$$L_o(p, v) = \frac{\rho}{\pi} E(p) \quad (8)$$

Ačkoli by v tomto případě bylo možné ukládat přímo výslednou barvu, osvětlení obvykle nevyžaduje tak vysokou frekvenci detailů jako textura albeda. Textury albeda navíc bývají opakovatelné, zatímco osvětlení může být v každém místě povrchu unikátní. To by znamenalo buď zvýšení rozlišení osvětlení na úroveň albeda včetně jeho opakování, aby byla zachována kvalita albeda, nebo naopak snížení rozlišení albeda na úroveň osvětlení. Ani jedna z možností však není optimální – buď by výrazně vzrostly požadavky na paměť, nebo by došlo ke snížení kvality albeda. V praxi se proto upřednostňuje uchovávat obě složky odděleně. (Akenine-Möller a kol. 2018, s. 474)

Průkopníkem předpočítaných hodnot intenzity záření, potažmo předpočítaného osvětlení jako takového, byly videohry od společnosti id Software *Quake* z roku 1996, který je využíval pro předpočítání přímého osvětlení, a posléze *Quake II* z roku 1997, který jej doplnil o nepřímé osvětlení a stal se tak první videohrou používající globální osvětlení (Akenine-Möller a kol. 2018, s. 474).

1.4.2 Směrové lightmapy

Předpočítání jediné „ploché“ intenzity záření bylo na konci 90. let 20. století poměrně běžné (Pettineo 2016 a), zejména kvůli omezeným schopnostem tehdejšího hardwaru. Ve skutečnosti však povrchy nejsou nikdy dokonale hladké. S pokrokem v grafických technologiích se začaly prosazovat i pokročilejší techniky, které umožnily simulovat drobné nerovnosti, jež ovlivňují, jak světlo interaguje s povrchem. Techniky pro simulování povrchových nerovností se souhrnně označují jako *bump mapping* (Akenine-Möller a kol. 2018, s. 208).

Jednu z nejběžnějších technik v této kategorii představuje *normal mapping* (Akenine-Möller a kol. 2018, s. 214), který spočívá v uložení detailních normál do textury – tzv. normálové mapy. Normály jsou zde uloženy v tečném prostoru, jež je vztažen vůči původní normále povrchu, což umožňuje znovupoužití téže textury na různě orientovaných površích (Akenine-Möller a kol. 2018, s. 209). Při zobrazování je normálová mapa namapována na příslušné povrchy a v každém bodě povrchu umožňuje vzorkovat upravenou detailní normálu, s jejíž pomocí vytváří iluzi jemných detailů a nerovností bez zvýšení geometrické složitosti modelu (Akenine-Möller a kol. 2018, s. 208).

Na počátku 21. století se normal mapping stal běžnou součástí grafických aplikací (Pettineo 2016 a), což pro předpočítané osvětlení představovalo novou výzvu. Předpočítání jediné „ploché“ hodnoty intenzity záření pro dané místo na povrchu se najednou jevilo jako nedostačující, neboť intenzita záření je vždy vztažena ke konkrétní normále povrchu.

Podobně jako by bylo možné předpočítat osvětlení včetně albeda, bylo by možné při předpočítání uvažovat i konkrétní normálu vzorkovanou z normálové mapy. V takovém případě by však rozlišení normál bylo limitováno frekvencí, při jaké je povrchové osvětlení předpočítáno. Normálové mapy však obvykle nabývají vyšší frekvence. To znamená, že má-li být normálová mapa udržována mimo předpočítané osvětlení, je při jeho předpočítání třeba brát v úvahu skutečnost, že normála v jakémkoli bodě na povrchu může nabývat různých směrů v rámci hemisféry nad osvětlovaným bodem.

Nově je tedy cílem zachytit směrovou závislost intenzity záření v daném bodě. Předpočítané osvětlení povrchu reagující na tento problém se obecně označuje jako směrové. Jeho podstata spočívá v tom, že namísto jediné hodnoty intenzity záření pro konkrétní povrchovou normálu se nyní ukládá celá distribuce hodnot intenzity záření v rámci hemisféry a tím zachycuje, jak se intenzita záření mění v závislosti na směru povrchové normály. Tento koncept lze zobecnit i pro potřeby light probes, kde je zapotřebí ukládat distribuci intenzity záření ze všech směrů okolo bodu – tedy v rámci celé koule. (Akenine-Möller a kol. 2018, s. 475)

Intenzitu záření lze předpočítat pouze pro povrchy, které odrážejí světlo podle lambertovské BRDF, neboť zůstává konstantní bez ohledu na směru pozorování. Pro povrchy využívajících pokročilejších BRDF, jejichž odraz se v závislosti na směru pozorování mění, je namísto toho třeba ukládat distribuci záře, jež přichází z daných směrů hemisféry nad osvětlovaným bodem (Akenine-Möller a kol. 2018, s. 393). Výsledný integrál dopadajícího osvětlení zahrnujícím jeho interakci s BRDF se pak dopočítává až při vykreslování a výpočtu osvětlení daného bodu (Akenine-Möller a kol. 2018, s. 493).

Z matematického hlediska jsou distribuce záře či intenzity záření funkcemi závislými na směru. K jejich popisu se proto využívá sférických funkcí, jež jsou definovány na povrchu jednotkové

koule. Sférická báze pak označuje soubor sférických funkcí, z jejichž lineárních kombinací lze složit libovolnou funkci na povrchu jednotkové koule. Z pohledu lightmap čili osvětlení na úrovni povrchu jsou užitečnými i hemisférické báze, které oproti sférickým definují hodnoty pouze na jedné polovině koule. (Akenine-Möller a kol. 2018, s. 393)

Přímočarou formou, jak zachytit směrovou závislost osvětlení by bylo využití tzv. *environment map*. Tyto speciální textury představují parametrizaci povrchu koule rozvinutého do 2D textury, kde každý texel uchovává zář přicházející z odpovídajícího směru. Základním předpokladem je, že zář závisí pouze na směru, což znamená, že světlo přichází z nekonečně vzdáleného zdroje a neovlivňují jej lokální překážky v prostředí. Přesnost této reprezentace je dána rozlišením textury. (Akenine-Möller a kol. 2018, s. 404–406, 393–394)

Při výpočtu osvětlení je nejprve nutné určit intenzitu záření. To znamená spočítat integrál, který odpovídá konvoluci příchozí záře s BRDF váženou kosinovým faktorem $(n \cdot l)^+$, jež se provádí přes hemisféru nad osvětlovaným bodem. Při konvoluci je příchozí zář vzorkována z environment mapy podle směru příchozího světla. Kvalita výsledného osvětlení je pak ovlivněna nejen rozlišením environment mapy, ale také počtem vzorků, které se z environment mapy odeberou při numerické aproximaci integrálu. S rostoucím počtem vzorků však roste i výpočetní náročnost.

Provádět takovou operaci opakovaně pro každý osvětlovaný bod by bylo výpočetně neúnosné. V praxi se proto přistupuje k různým optimalizacím, které umožňují tento výpočet výrazně urychlit předpočítáním výsledků konvoluce (Akenine-Möller a kol. 2018, s. 424–427).

Lambertovská BRDF, která je konstantní v každém směru, může být vytknuta z integrálu. V takové podobě integrál odpovídá konvoluci záře s kosinovým faktorem $(n \cdot l)^+$ neboli výpočtu intenzity záření, která je ovlivněna pouze směrem příchozího světla a normálou povrchu. Výsledná intenzita záření tedy v případě dané normály zůstává konstantní. V praxi se proto využívá tzv. *irradiance environment mapy*, která uchovává intenzitu záření, jež je indexována podle povrchové normály. Jinými slovy jde o ekvivalent základní environment mapy, která namísto záře ukládá intenzitu záření. (Akenine-Möller a kol. 2018, s. 424–427)

Konvoluce záře s kosinovým faktorem působí jako filtr s dolní propustí, který přirozeně potlačí vysokofrekvenční složky. V důsledku toho se intenzita záření mění hladce a plynule v závislosti na povrchové normále. Irradiance environment mapy jsou tak ve srovnání se září z původní environment mapy značně rozmazané, a proto je lze uložit v podstatně nižším rozlišení bez výrazné ztráty kvality. (Akenine-Möller a kol. 2018, s. 424–427)

Použití irradiance environment map za běhu je efektivní, neboť pro zjištění intenzity záření stačí vzorkovat texel odpovídající povrchové normále. Samotné generování irradiance environment map je však náročné jak výpočetně, tak paměťově, protože vyžaduje zpracování velkého množství vzorků původní záře a následné uchování výsledných hodnot pro každý směr zvlášť.

Ačkoli několik předchozích odstavců popisovalo koncept environment map, jejich představení v kontextu lightmap sloužilo zejména k intuitivnímu pochopení jejich funkce, neboť tvoří základ pro pokročilejší techniky, jež budou představeny dále. Jejich využití v kombinaci s lightmapami je však spíše teoretickou úvahou, která by byla v praxi příliš neefektivní. Vezmeme-li v potaz, že nejmenší adresovatelnou jednotkou lightmapy je texel, znamenalo by to nutnost ukládat v každém texelu celou další texturu, což by vedlo k enormním paměťovým nárokům.

Přesto samotné environment mapy stále nacházejí uplatnění v počítačové grafice, například v technikách jako je *reflection mapping* (Akenine-Möller a kol. 2018, s. 404–406) nebo obecně tam, kde je třeba uchovávat přichozí zář s vysokou přesností, což není případ lightmap.

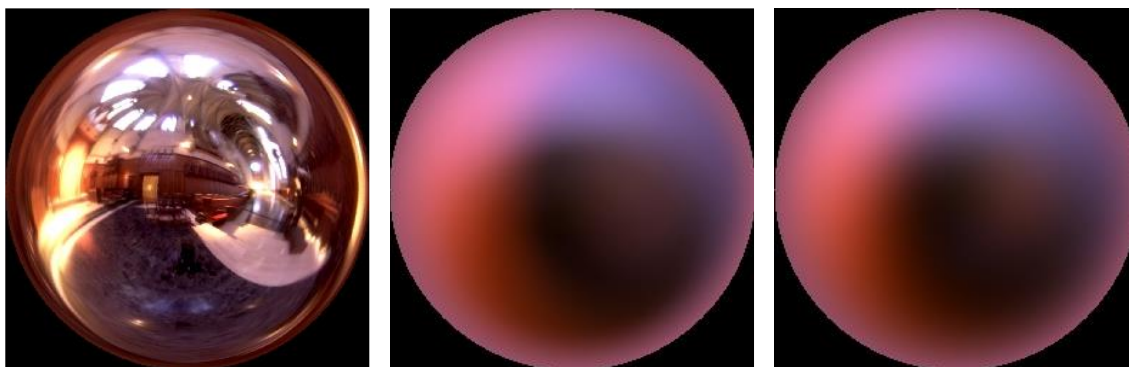
Skutečnost, že se intenzita záření mění hladce a plynule v závislosti na povrchové normále, byla klíčovým poznatkem, který vedl ke zjištění, že irradiance environment mapy lze efektivně reprezentovat pomocí *sférických harmonických funkcí* (SH) (Ramamoorthi a Hanrahan 2001).

Pomocí SH lze obecně aproximovat libovolnou funkci definovanou na povrchu jednotkové koule. Podobně jako *Fourierova transformace* rozkládá signál do lineární kombinace sinových a kosinových funkcí s různými váhami, SH umožňují rozložit funkci definovanou na povrchu jednotkové koule do lineární kombinace jednodušších báзовých funkcí (Green 2003 s. 8–9, 14). Podrobným popisem SH se zabývá např. Green (2003), tato práce se však soustředí především na jejich praktické využití.

Proces aproximace původní funkce se označuje jako *projekce* a spočívá ve výpočtu koeficientů, které udávají nakolik se původní funkce podobá daným báзовým funkcím. To se děje integrací součinu hodnoty báзовé funkce a hodnoty původní funkce po celém jejím definičním oboru. Aproximaci původní funkce pak lze rekonstruovat jako součet všech báзовých funkcí vážených odpovídajícími koeficienty. (Green 2003 s. 8–9, 14)

Přesnost této aproximace je dána počtem použitých báзовých funkcí, resp. řádem SH. Čím vyšší řád použijeme, tím více detailů původní funkce lze zachytit, ovšem za cenu vyššího počtu potřebných koeficientů.

Možnost využití SH jako kompaktní a efektivní reprezentace irradiance environment map byla poprvé představena Ramamoorthim a Hanrahanem (2001). Jejich práce demonstruje, že irradiance environment mapy lze reprezentovat s přesností přibližně 1 % pomocí pouhých devíti koeficientů, resp. SH třetího řádu (Ramamoorthi a Hanrahan 2001).



Obr. 11: Environment mapa „Grace Cathedral“ uchovávající zář (vlevo) a jí odpovídající irradiance environment mapa uchovávající intenzitu záření reprezentovaná pomocí textury (uprostřed) a pomocí sférických harmonických funkcí třetího řádu (vpravo)

Zdroj: levý obrázek – Debevec (1998), prostřední a pravý obrázek – Ramamoorthi a Hanrahan (2001)

Z toho vyplývá, že environment mapu či irradiance environment mapu, resp. distribuci záře či intenzity záření, lze namísto explicitního ukládání hodnot osvětlení pro každý směr reprezentovat jen několika málo koeficienty.

SH však přinášejí nejen úsporu paměťovou, ale i výpočetní. Vzhledem k tomu, že SH reprezentují environment mapu ve frekvenční oblasti, výpočet konvoluce s jinou se výrazně zjednoduší.

Mějme dvě funkce definované na povrchu koule reprezentované pomocí koeficientů SH, jejich konvoluce odpovídá skalárnímu součinu odpovídajících koeficientů obou funkcí. (Green 2003 s. 19)

To znamená, že zatímco tradiční výpočet irradiance environment map vyžaduje $O(T \cdot S)$ operací, kde T je počet texelů v cílové irradiance environment mapě a S je počet texelů ve vstupní environment mapě, v případě výpočtu koeficientů SH jde o $O(K \cdot S)$ operací, kde K je počet bazových funkcí. Díky tomu je tato metoda přibližně T/K -krát rychlejší než tradiční postup. (Ramamoorthi a Hanrahan 2001)

Ramamoorthi a Hanrahan (2001) zároveň uvádějí i koeficienty pro konvoluci záře s kosinovým faktorem $(n \cdot l)^+$, pomocí kterých lze z koeficientů záře efektivně vyjádřit intenzitu záření v libovolném směru. To znamená, že po vyjádření konkrétního směru pomocí koeficientů SH lze intenzitu záření v daném směru získat skalárním součinem těchto koeficientů s konvolučními koeficienty a koeficienty záře.

SH jsou jednou z nejpoužívanějších metod pro zachycení směrové závislosti osvětlení. Lze je využít jak sféricky, tak hemisféricky, tedy jak pro light probes, tak pro lightmapy. V praxi se nejčastěji využívají SH druhého nebo třetího řádu. Třetí řád využívá devět koeficientů na barevný kanál a nabízí téměř dokonalou kvalitu, avšak za cenu vyšší paměťové náročnosti. Druhý řád využívá pouze čtyři koeficienty na barevný kanál, ale kvalita osvětlení je nižší, protože nezvládá zachytit jemné detaily. Druhý řád se tak hodí např. pro nepřímé osvětlení, které obvykle nevyžaduje vysokou kvalitu. (Akenine-Möller a kol. 2018, s. 475, s. 427–428)

S porozuměním SH lze nyní popsat, jakým způsobem se SH využívají v kontextu lightmap. Zatímco environment mapy jsou příliš objemné na to, aby bylo praktické je ukládat pro každý jednotlivý texel lightmapy, SH představují výrazně úspornější a přijatelnou alternativu.

V praxi to znamená, že pro každý texel je potřeba uložit několik koeficientů odpovídajících danému řádu SH. Z předchozího textu je však patrné, že devět koeficientů je optimální maximum. Každý koeficient je tvořen třemi barevnými kanály. Paměťové nároky tak oproti „plochým“ lightmapám vzrostou úměrně počtu koeficientů. Přestože lightmapy využívají SH pouze hemisféricky, na počet koeficientů to nemá vliv. Jediným rozdílem je, že hemisféra odvrácená vůči povrchové normále přirozeně nabývá nulových hodnot.

Byť se pojem „texel tvořený devíti koeficienty“ může zdát jako abstraktní, ve skutečnosti se jedná o devět samostatných textur. Případně si lze tuto skutečnost představit jako devítivrstvou texturu. Stále však hovoříme o jedné směrové lightmapě.

V roce 2003 spatřila světlo světa videohra *Max Payne 2* od společnosti Remedy Entertainment, která je pravděpodobně jednou z prvních videoher, jež využívá SH druhého řádu k osvětlování dynamických objektů pomocí irradiance volume. Tak uvádí Mitchell a kol. (2006, s. 133) s odkazem na osobní komunikaci s Jaakko Lehtinenem.

Další přelomový okamžik představuje v roce 2004 vydání videohry *Half-Life 2* od společnosti Valve, která využívá jak irradiance volume, tak i směrových lightmap, přičemž k aproximaci směrové závislosti záře využívají vlastních bází.

Formát pro uchovávání směrových lightmap je označován jako *Radiosity Normal Mapping*, jehož název je odvozen z algoritmu, který Valve používalo k výpočtu lightmap a techniky normal mappingu. (Mitchell a kol. 2006, s. 131–132)

Radiosity Normal Mapping využívá jednoduchou hemisférickou bázi, která je definována třemi ortogonálními směry. Kvůli svému použití ve videohře Half-Life 2 se dnes tato báze rovněž označuje jako *HL2 basis*. Záře je pak aproximována třemi koeficienty, které se získají vzorkováním záře v každém ze tří směrů báze. (Mitchell a kol. 2006, s. 131–132) (Akenine-Möller a kol. 2018, s. 402–404)

Druhou, tentokrát sférickou, bázi, kterou představil Half-Life 2 je *Ambient Cube*. Ta ukládá zář jako šest koeficientů vzorkovaných podle hlavních pozitivních a negativních os X, Y, Z. Tento přístup nabízí podobnou kvalitu jako druhý řád SH, ale s nižšími výpočetními nároky. (Mitchell a kol. 2006, s. 133–135) (Akenine-Möller a kol. 2018, s. 478)

Směrové lightmapy, jak je poprvé využil Half-Life 2, představovaly přelomový moment, který ukázal výhody aproximace směrové závislosti záře. Postupem času však došlo k dalšímu vývoji směrových reprezentací osvětlení, kdy začaly vznikat nové metody snažící se vyvážit kvalitu, výpočetní náročnost a paměťové požadavky.

Jednou z hemisférických alternativ je tzv. *H-basis*. Díky hemisférické reprezentaci dosahuje lepšího poměru mezi kvalitou a paměťovou náročností. Pro dosažení kvality odpovídající třetímu řádu sférických harmonických postačuje pouze šest koeficientů. H-basis vyžaduje správnou orientaci vůči povrchu, což je zajištěno použitím lokálního tečného prostoru. (Akenine-Möller a kol. 2018, s. 475)

Ambient dice je pokročilejší variantou Ambient cube, která využívá dvanácti hodnot orientovaných podle vrcholů dvacetistěnu. Tato metoda nabízí kvalitu srovnatelnou s třetím řádem SH. (Akenine-Möller a kol. 2018, s. 476–478)

Jinou, poměrně intuitivní reprezentací je tzv. *AHD basis* (*Ambient/Highlight/Direction*), která popisuje sférickou funkci pomocí tří složek – ambientního světla⁵ (A), barvy směrového světla (H) a směru směrového světla (D). Tento způsob byl poprvé využit již roku 1999 ve videohře *Quake III* od společnosti id Software, a to v rámci tzv. *light volumes*, které slouží k osvětlování dynamických objektů (Hoover 2016). Stejný způsob byl později použit i ve videoherní sérii *Call of Duty* od společnosti Activision nebo ve videohře *The Last of Us* z roku 2013 od společnosti Naughty Dog (Iwanicki 2013, s. 14–17). (Akenine-Möller a kol. 2018, s. 476)

Další metodou jsou *sférické Gaussovy funkce*, které umožňují zachytit směrovou distribuci světla s vysokou přesností. Oproti ostatním metodám tento formát nepracuje s intenzitou záření, ale se září. Díky vysoké přesnosti je tento formát vhodný i pro spekulární osvětlení. (Akenine-Möller a kol. 2018, s. 477)

⁵ Rozptýlené světlo přicházející z okolí ve všech směrech, bez konkrétního směru nebo zdroje.

1.5 Path tracing

Path tracing je jednou z výpočetních metod pro simulaci globálního osvětlení, která byla představena Jamesem T. Kajiyou (1986) společně se zobrazovací rovnicí.

Jedna ze základních variant – jednosměrný zpětný path tracing (*unidirectional backward path tracing*) – spočívá ve vržení paprsku z každého pixelu virtuální kamery a sledování přenosu světla při jeho průchodu scénou zpět k možným zdrojům světla⁶.

Následující popis modeluje interakci povrchu pouze pomocí BRDF, ale stejný princip lze aplikovat i na obecnější BSDF.

Sledování cesty paprsku začíná vržením tzv. primárního paprsku z pixelu kamery. Při dopadu paprsku na povrch dojde k interakci, která se řídí jeho materiálovými vlastnostmi, resp. jeho BRDF. Část světla může být pohlcena a zbylá část se odrazí novým směrem jako tzv. sekundární paprsek. Při každé interakci paprsku s povrchem se akumuluje přímé světlo emitované povrchem a světlo odražené z jiných povrchů. Tento proces se provádí opakovaně, dokud paprsek nezanikne interakcí s prostředím (například oblohou nebo vzdáleným pozadím) nebo není-li dosaženo maximálního počtu odrazů. (Boksansky a Marrs 2021, s. 174-175)

Povšimněte si, že tento základní algoritmus striktně nerozlišuje zdroje světla. Vždy se jedná o povrchy, které mohou emitovat světlo. To je dáno z podstaty simulovat osvětlení podle fyzikálně založeného modelu, kdy je světlo vyzařováno z povrchu o určité ploše (Pharr a kol. 2017, s. 8). Bezrozměrné zdroje světla s nulovou plochou tak do tohoto modelu zcela nezapadají a vyžadují individuální zacházení. Tato problematika je podrobněji rozebrána při popisu explicitního vzorkování zdrojů světla v rámci kapitoly 1.5.2.

Pro úplnost je vhodné zmínit, že v reálném světě je světlo šířeno uvnitř médií (*participating media*), která se podílejí na přenosu světla. Mezi ta patří např. plyny, mlha, prach a další látky, jež mohou světlo pohlcovat či rozptylovat. Pro fyzikálně přesnou simulaci osvětlení je tedy třeba zohlednit i tyto efekty. V této práci je však za médium implicitně považováno vakuum a jeho účinky jsou tak zanedbány. (Pharr a kol. 2017, s. 13–16)

Z technického hlediska je zásadním úkolem algoritmu nejen samotný výpočet interakce světla s povrchy, ale také efektivní hledání průsečíků paprsků se scénou. Při vržení paprsku je cílem nalézt průsečík s povrchem, který je nejbližší počátku paprsku. Kromě samotné polohy průsečíku je pro výpočet osvětlení nutné znát i jeho BRDF, normálu a případně i další atributy v závislosti na použité metodě stínování⁷. Je-li geometrie dána trojúhelníkovou sítí, atributy v průsečíku jsou dány interpolací atributů uložených ve vrcholech zasaženého trojúhelníku.

Scéna je typicky tvořena velkým množstvím objektů. Naivním způsobem by bylo ověřovat průnik paprsku se všemi objekty. U velkých scén by to znamenalo enormní objem práce, který by celý výpočet zásadně zpomaloval. V praxi je proto efektivnější využívat prostorových akceleračních struktur, které jsou při nalézání průsečíku schopny rychle a včasné eliminovat větší celky scény. (Pharr a kol. 2017, s. 8)

⁶ Možnost zpětného sledování dráhy světla vychází z Helmholtzovy reciprocity, která umožňuje směr šíření světla obrátit, aniž by došlo ke změně přenosu energie.

⁷ Mezi další atributy mohou patřit např. tangenta, UV souřadnice textury albeda a jiné.

1.5.1 Monte Carlo integrování

Protože analytické řešení zobrazovací rovnice je prakticky nemožné, path tracing využívá *Monte Carlo integrování*, které umožňuje výpočet aproximovat pomocí stochastického vzorkování.

Monte Carlo integrování je založeno na numerickém odhadu integrálu funkce. Základní myšlenka spočívá v tom, že chceme-li spočítat určitý integrál, můžeme jej aproximovat jako průměr hodnot náhodně vybraných bodů v daném definičním oboru.

V kontextu path tracingu to znamená, že vzorkujeme-li směry odrazu světla dostatečně mnohokrát, získáme statisticky správnou hodnotu, udávající množství světla dopadajícího na daný pixel.

V Monte Carlo metodách hraje klíčovou roli vzorkování. Základním přístupem je vzorkování s konstantní hustotou pravděpodobnosti (PDF, *probability density function*), resp. s rovnoměrným rozdělením pravděpodobnosti, kdy všechny vzorky nabývají stejné pravděpodobnosti, že budou zvoleny.

Matematicky lze Monte Carlo integrování vyjádřit jako:

$$I = \int_{\Omega} f(x) d\mu(x) \quad (9)$$

kde $f(x)$ je integrovaná funkce a Ω je oblastí nad kterou integrujeme.

Tento integrál můžeme aproximovat pomocí náhodného vzorkování:

$$F_N = \frac{1}{N} \sum_{i=1}^N f(X_i) \quad (10)$$

kde X_i jsou nezávislé a rovnoměrně vzorkované hodnoty z definičního oboru a N je počet vzorků.

Tento odhad vychází ze zákona velkých čísel, který zaručuje, že s rostoucím N pak F_N konverguje k přesnějšímu odhadu integrálu.

Tento vzorec však není vždy efektivní, zejména pokud některé oblasti integrace přispívají k výsledku více než jiné. Proto se často místo rovnoměrného vzorkování používá vzorkování s jinou PDF $p(x)$, kde jsou některé oblasti preferovány. V takovém případě je nutné každý vzorek vážit dle své PDF.

$$F_N = \frac{1}{N} \sum_{i=1}^N \frac{f(X_i)}{p(X_i)} \quad (11)$$

V path tracingu je nutné generovat náhodné vzorky z různých oblastí. První oblastí jsou pixely na kameře, které slouží jako počátek pro primární paprsky.

Osvětlení, které počítáme, však není aproximací osvětlení přichozícího do jednoho bodu, nýbrž na plochu celého pixelu. Byť by bylo možné naivně vzorkovat osvětlení ve středu pixelu, při nedostatečném rozlišení by takový přístup vedl k aliasingu. Namísto toho se v praxi využívá stochastické vzorkování pixelů, kdy se primární paprsky vrhají z náhodně zvolených bodů uvnitř pixelu.

Druhou oblastí vzorkování jsou směry paprsků, které určují směr odraženého paprsku, pro nějž budeme vyhodnocovat osvětlení. Uvažujeme-li pouze BRDF, tento směr se volí v rámci hemisféry nad osvětlovaným bodem povrchu.

Avšak jestliže jsou směr paprsku a jeho interakce s materiálem náhodné, každý vzorek přispívá do výsledku jinou hodnotou. To vede ke vzniku rozptylu, který vyjadřuje, jak moc se jednotlivé vzorky liší od skutečné očekávané hodnoty. Vysoký rozptyl se pak na výsledném obrazu projevuje jako vizuální šum.

1.5.2 Metody pro snížení rozptylu odhadu

Přestože path tracing poskytuje vysoce realistické výsledky, šum zůstává jeho zásadní nevýhodou. Základním způsobem, jak snížit šum, je navýšit počet vzorků, čímž se postupně průměrují náhodné odchylky jednotlivých vzorků. Bohužel, zvýšení počtu vzorků výrazně zvyšuje výpočetní náročnost, což není vždy praktické.

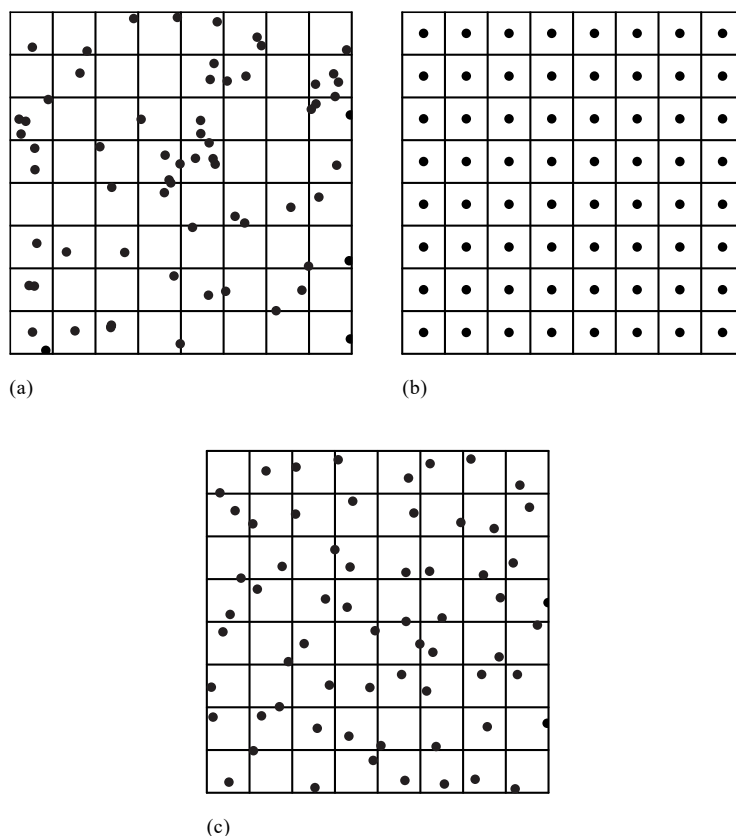
Proto se využívají různé metody pro snížení rozptylu odhadu (*variance reduction techniques*), které umožňují dosáhnout kvalitnějších výsledků i při nižších počtech vzorků.

Jedna z kategorií metod pro snížení rozptylu je založena na **důmyslném umístění vzorků** tak, aby jejich rozložení lépe pokrývalo vlastnosti integrované funkce a minimalizovalo se riziko, že některé z podstatných rysů nebudou zachyceny. (Pharr a kol. 2017, s. 789)

Mezi běžné strategie v této kategorii patří *stratifikované vzorkování*, které rozděluje integrační oblast do několika navzájem nepřekrývajících se podoblastí – *strat*. Při vzorkování se pak odebírají vzorky z každé ze *strat*, což snižuje pravděpodobnost shlukování více vzorků ve stejných oblastech. (Pharr a kol. 2017, s. 789–790)

Ideální je, pokud vzorky nejsou umístěny pravidelně. Toho lze docílit tak, že v každé oblasti je náhodně umístěn vzorek takovým způsobem, že se střed daného strata mírně posune až o polovinu své šířky a výšky. (Pharr a kol. 2017, s. 432)

Náhodný posun narušuje pravidelnost vzorků, čímž namísto nežádoucího aliasingu vzniká šum, který je vizuálně přijatelnější.



Obr. 12: Tři způsoby vzorkování ve 2D: (a) náhodné vzorkování, (b) pravidelně stratifikované vzorkování, (c) stratifikované vzorkování

Zdroj: Pharr a kol. 2017, s. 435)

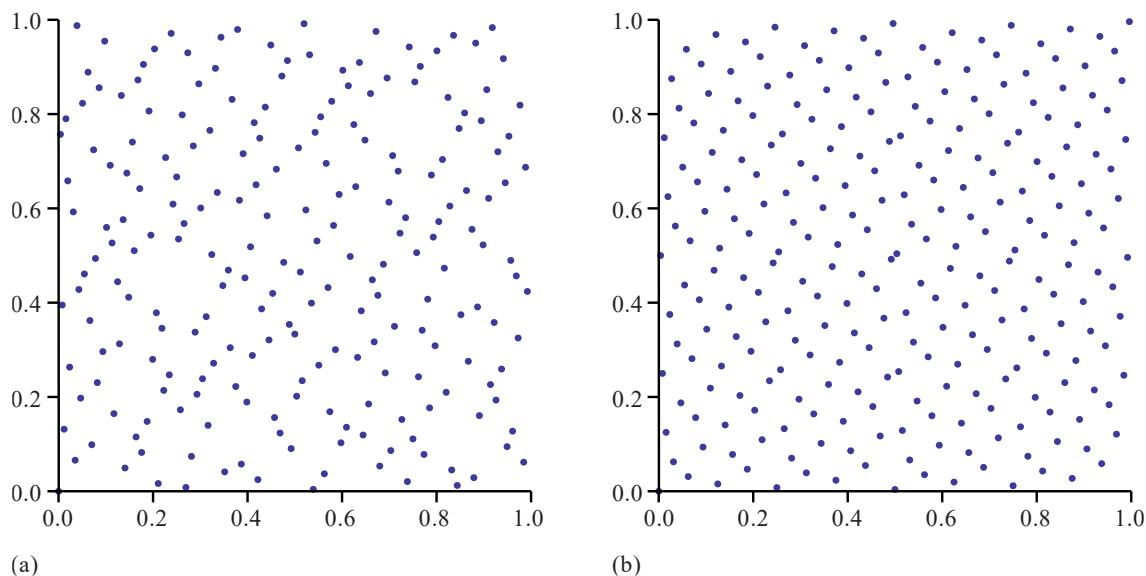
Stratifikované vzorkování nikdy nezvyšuje rozptyl a obvykle jej výrazně snižuje. Jeho účinnost však závisí na tom, nakolik stratifikace odpovídá frekvenci změn integrované funkce. Hlavní nevýhodou je, že počet potřebných vzorků roste exponenciálně s počtem rozměrů vzorkovaného prostoru. To znamená, že při plné stratifikaci prostoru o D rozměrech a použití S strat na každý rozměr, by celkový počet vzorků byl roven S^D . To výrazně zvyšuje výpočetní náročnost, která může dospět do stavu, kdy už výpočet nemusí být prakticky realizovatelný. (Pharr a kol. 2017, s. 789–792)

Další ze strategií je *Kvazi-Monte Carlo integrování*, které spočívá v nahrazení generátorů pseudonáhodných čísel, používaných při vzorkování ve standardním Monte Carlu, deterministickými posloupnostmi s nízkou diskrepancí (Pharr a kol. 2017, s. 792). Tyto posloupnosti jsou konstruovány tak, aby rovnoměrně pokrývaly integrační oblast (Pharr a kol. 2017, s. 417).

Diskrepance vyjadřuje, jak rovnoměrně jsou vzorky rozmístěny v n -rozměrném prostoru $[0,1]^n$ tím, že porovnává objem různých oblastí v tomto prostoru s podílem vzorků, které do dané oblasti spadají. Ideálně by měl podíl vzorků v dané oblasti odpovídat jejímu relativnímu objemu vůči celkovému objemu prostoru. (Pharr a kol. 2017, s. 417–419)

Mezi zástupce posloupností s nízkou diskrepancí patří např. Haltonova nebo Hammersleyho posloupnost. Podstatný technický rozdíl mezi nimi spočívá v tom, že zatímco z Haltonovy posloupnosti lze odebírat vzorky bez předchozí znalosti jejich celkového počtu, u Hammersleyho posloupnosti je tato znalost nezbytná (Pharr a kol. 2017, s. 443–444). To v praxi znamená, že

Haltonova posloupnost umožňuje progresivní vzorkování, které je užitečné tehdy, nechceme-li se omezovat stanovením skutečného počtu potřebných vzorků předem, nýbrž chceme být schopni generovat vzorky průběžně podle aktuálních požadavků, například s ohledem na rozptyl.



Obr. 13: Několik prvních bodů posloupností s nízkou diskrepancí ve 2D: (a) Haltonovy posloupnosti – 216 bodů a (b) Hammersleyho posloupnosti – 256 bodů

Zdroj: Pharr a kol. 2017, s. 445)

Další z kategorií metod pro snížení rozptylu představuje *vzorkování podle důležitosti* (*importance sampling*), které cíleně upřednostňuje ty vzorky, které mají větší dopad na výslednou hodnotu. Známe-li předem PDF vzorkované funkce, můžeme vzorkování přizpůsobit tak, aby odpovídalo tvaru této funkce. V takovém případě již neplatí, že všechny vzorky nabývají stejné hodnoty PDF, a proto je při Monte Carlo integrování nutné každý vzorek vážit podle jeho PDF. (Pharr a kol. 2017, s. 794)

Za základní aplikaci vzorkování podle důležitosti lze považovat tzv. vzorkování hemisféry vážené kosinem (*cosine-weighted hemisphere sampling*). To znamená, že při výběru vzorku přihlížíme ke členu $(n \cdot l)$, nacházejícím se v rovnici odrazu, který odpovídá kosinu úhlu mezi normálou a směrem generovaného vzorku. Tento způsob vzorkování má za následek, že namísto naivního rovnoměrného vzorkování hemisféry, je přikládána vyšší váha směrům, které jsou blíže normále povrchu. (Lindqvist 2021, s. 328)

Tuto metodu lze dále rozšířit na vzorkování podle materiálu, tedy podle BRDF s ohledem na člen $f(l, v)(n \cdot l)$.

Chování dokonale difuzního materiálu s lambertovskou BRDF odpovídá vzorkování hemisféry vážené kosinem. Nicméně např. u lesklejších materiálů je vhodnější častěji vzorkovat směry blízké zrcadlovému odrazu.

V případě lambertovské BRDF navíc vzorkování hemisféry vážené kosinem přináší další přívětivou vlastnost, která umožňuje zjednodušit výpočet odhadu integrálu.

Uvážíme-li, že lambertovská BRDF je definována jako:

$$f(l, v) = \frac{\rho}{\pi} \quad (12)$$

a PDF pro vzorkování v hemisféře je volena jako:

$$p(l) = \frac{(n \cdot l)}{\pi} \quad (13)$$

pak po dosazení do vzorce dostáváme:

$$L_o(p, v) \approx \frac{1}{N} \sum_{i=1}^N \frac{\frac{\rho}{\pi} L_i(p, l)(n \cdot l)}{\frac{(n \cdot l)}{\pi}} \quad (14)$$

Protože se členy $(n \cdot l)$ a $\frac{1}{\pi}$ v čitateli i jmenovateli navzájem pokrátí, výpočet odhadu integrálu se zjednoduší na:

$$L_o(p, v) \approx \frac{1}{N} \sum_{i=1}^N \rho L_i(p, l) \quad (15)$$

Další běžnou strategií je vzorkování podle zdrojů světla, v odborné literatuře též nazývaných jako *explicit light sampling* nebo *next event estimation* (NEE). NEE spočívá v tom, že se v každém bodě interakce vzorkuje přímo směr ke světelnému zdroji, resp. k některému bodu na jeho ploše, a vyhodnotí se jeho příspěvek k výslednému osvětlení. Tento směr se obvykle označuje jako stínový paprsek (*shadow ray*), neboť jeho cílem je také ověřit viditelnost/zastínění světelného zdroje vůči osvětlovanému bodu. (Boksansky a Marrs 2021, s. 181)

Tato strategie je zásadní zejména pro světelné zdroje s malou plochou. Pokud bychom se spoléhali pouze na náhodné vzorkování hemisféry, pravděpodobnost, že paprsek zasáhne malý světelný zdroj, by byla nízká. V důsledku toho by velké množství paprsků přispělo nulovou hodnotou, což by nejen zpomalilo konvergenci, ale zároveň by vedlo k plýtvání výpočetním výkonem.

Navíc u bezrozměrných světelných zdrojů s nulovou plochou je šance jejich náhodného zasažení prakticky nulová. To znamená, že bez využití NEE by při path tracingu nebylo použití takových světél vůbec možné.

Ne vždy je však praktické vrhat stínové paprsky ke všem světelným zdrojům ve scéně. Jedním z jednoduchých řešení je namísto všech světelných zdrojů náhodně zvolit pouze jeden z nich s konstantní PDF $1/N$ kde N je celkový počet světelných zdrojů ve scéně. Tato metoda však často způsobuje šum, protože nezohledňuje důležitost, resp. vlastnosti jednotlivých světél, jako je jejich intenzita, velikost nebo vzdálenost zvoleného světelného zdroje od osvětlovaného bodu. (Boksansky a Marrs 2021, s. 183)

Alternativním, avšak výpočetně náročnějším přístupem by bylo vzorkování světelných zdrojů podle jejich skutečného příspěvku k výslednému osvětlení. To by znamenalo nejprve vypočítat interakci každého světla s povrchovou BRDF, na jejímž základě by bylo možné určit PDF jednotlivých zdrojů světla. Tato znalost by pak umožnila použít vzorkování podle důležitosti, které by upřednostňovalo zdroje světla s větším vlivem na výsledné osvětlení. (Boksansky a Marrs 2021, s. 183)

Praktickým kompromisem mezi těmito přístupy je namísto výpočtu interakce s povrchovou BRDF pro všechny zdroje světla uvažovat pouze podmnožinu kandidátů (např. osm), vybraných

rovnoměrným vzorkováním. Pro vybrané zdroje světla se následně vypočítá celá nebo částečná BRDF a jeden z nich je náhodně zvolen a vážen jeho PDF odpovídající jeho skutečnému příspěvku k osvětlení. (Boksansky a Marrs 2021, s. 184)

V praxi však není vhodné spoléhat pouze na vzorkování podle materiálu nebo pouze na vzorkování podle zdrojů světla, protože jejich účinnost závisí na konkrétní kombinaci povrchu a světelného zdroje. (Veach 1998, s. 256–257)

Pro dosažení optimálních výsledků je proto vhodné obě metody kombinovat. Erich Veach ve své disertační práci představil tzv. *multiple importance sampling* (MIS), které kombinuje oba přístupy s cílem minimalizovat rozptyl a zároveň využít výhod obou metod. (Veach 1998, s. 251)

MIS obecně umožňuje kombinovat různé strategie vzorkování podle důležitosti a dynamicky jim přiřadit odpovídající váhy podle toho, jak daná strategie vyhovuje aktuální situaci. Cílem však není zvolit tu lepší z nich, ale všechny je zkombinovat tak, aby byly zachovány jejich silné stránky. (Veach 1998, s. 251–252)

Mezi další metody používané pro snížení výpočetní náročnosti Monte Carlo integrování patří i metoda zvaná *ruská ruleta*, která umožňuje předčasně ukončit sledování paprsku v okamžiku, kdy jeho další odrazy představují zanedbatelný příspěvek k výslednému osvětlení, neboť vlivem předchozích odrazů již došlo ke ztrátě převážné části přenášené energie. (Boksansky a Marrs 2021, s. 179)

V praxi to funguje tak, že před sledováním každého sekundárního paprsku se náhodně rozhodne, zda v jeho sledování pokračovat. Nejprve se vygeneruje náhodné číslo ξ v intervalu $(0, 1]$. Je-li $q \leq \xi$, sledování paprsku se ukončí a jeho příspěvek je nahrazen nulou. Pokud naopak $q > \xi$, sledování paprsku pokračuje, přičemž jeho následný příspěvek se násobí vahou $1/(1-q)$, která kompenzuje vynechání některých vzorků. (Pharr a kol. 2017, s. 787–788)

Aby se zabránilo předčasnému ukončování světelných drah, které by mohlo vést ke ztrátě důležitých vzorků, lze použití ruské rulety odložit a aplikovat je až po několika počátečních odrazech. (Boksansky a Marrs 2021, s. 179)

Mimo různé vzorkovací strategie, které se snaží o redukci šumu ve výsledném obraze, se v praxi často využívají také *denoisingové filtry*. Mezi základní vyhlazovací techniky patří například *bilaterální filtr* nebo *À-Trous filtr*. Oba tyto filtry jsou využívány herním enginem *Unity* pro denoising lightmap (Unity Technologies nedatováno). À-Trous filtr byl rovněž použit v herním enginu *Frostbite* (Apers a kol. 2019, s. 16).

Vedle těchto metod však existují i pokročilejší denoisingové metody založené na strojovém učení. Mezi ty patří například *Intel Open Image Denoise* (Intel nedatováno c) nebo *NVIDIA OptiX Autoencoder* (NVIDIA nedatováno b). V kontextu lightmap je Intel Open Image Denoise využíván například herním enginem *Godot* (Linietsky a kol. nedatováno). Herní engine *Unity* pak umožňuje použití obou těchto filtrů (Unity Technologies nedatováno).

2 Návrh řešení

Následující část se zaměřuje na návrh a strukturu řešení mého nástroje pro generování lightmap, včetně použitých technologií a klíčových komponent implementace. Nástroj nese název *Illux*, avšak dále je v textu označován jednoduše jako *nástroj*.

Cílem práce je implementovat nástroj, který umožní výpočet statického osvětlení ve formě lightmap pro uživatelem zadanou scénu, přičemž výstupem má být scéna s výslednými lightmapami a přidruženými UV souřadnicemi pro jejich správné mapování na původní geometrii scény.

Jelikož v praxi je takový nástroj typicky součástí větších grafických aplikací jako je herní engine či modelovací software, při návrhu řešení a identifikaci požadovaných funkcí jsem se snažil vycházet z potřeb, které vyžadují videohry a jejich designéri.

Jednou z klíčových vlastností takového nástroje je nejen samotné generování lightmap, ale také možnost interakce a kontroly průběhu procesu generování. Byť by bylo možné nástroj implementovat zcela bez uživatelského rozhraní, je praktické poskytnout alespoň základní vizualizaci scény včetně průběhu výpočtu a umožnit konfiguraci vstupních parametrů, což usnadní testování a ladění výsledků. To díky lepší kontrole nad jednotlivými fázemi procesu usnadní nejen vývoj samotného nástroje, ale zároveň zpříjemní uživatelskou zkušenost.

Dalším důležitým faktorem je schopnost poskytnutí rychlé zpětné vazby uživateli o aktuálním pokroku výpočtu. Samotný výpočet může mnohdy trvat řádově minuty či hodiny v závislosti na složitosti geometrie a zvolených vstupních parametrech, jež ovlivňují kvalitu výstupu. Naprosto základním ukazatelem by bylo zobrazení zbývajících množství práce v procentech vůči celkovému množství. To však uživateli nedá žádnou představu o tom, jak bude vypadat výsledný výstup. Proto je vhodné, aby navržené řešení umožňovalo progresivní výpočet lightmap. To znamená, že uživatel má možnost průběžně sledovat mezivýsledky, kdy se globální osvětlení v lightmapě postupně zpřesňuje. Takto může v reálném čase posoudit kvalitu průběžně počítaného osvětlení, identifikovat případné artefakty či nedostatky v raných fázích výpočtu a v případě potřeby upravit parametry, aniž by musel čekat na dokončení celého procesu. Díky tomuto přístupu lze rychleji iterovat nad možnými řešeními, což urychluje hledání optimální konfigurace a výrazně zjednodušuje proces ladění.

Progresivní výpočet je tedy jedním z důvodů, proč jsem pro výpočet globálního osvětlení zvolil path tracing. Jeho vhodnost je dána povahou Monte Carlo integrování, které pracuje na základě vzorkování osvětlení pomocí velkého množství náhodně vržených paprsků. Každý průchod výpočtu přidává nové vzorky, což postupně zpřesňuje výsledek.

Path tracing se dále vyznačuje poměrně přímočarou a intuitivní implementací, protože přímo vychází z fyzikálních principů šíření světla. Každý paprsek je sledován nezávisle a jeho průchod scénou se řídí fyzikálními zákony, což umožňuje dosažení vysoce realistických výsledků. Nicméně kvůli náhodnému vzorkování se výsledky pouze přibližují realitě a vždy obsahují určitou úroveň šumu. Pro dosažení vizuálně uspokojivé kvality je proto zapotřebí dostatečného množství vzorků, které pomáhají šum redukovat. Šum však lze redukovat i pomocí jiných metod pro snížení rozptylu odhadu představených v kapitole 1.5.2.

V neposlední řadě je path tracing také flexibilní s ohledem na konkrétní požadavky osvětlovacího modelu pro danou aplikaci. To znamená, že jej lze snadno rozšířit o podporu různých BSDF, typů světelných zdrojů či efektů jako jsou stíny, průsvitnost nebo podpovrchový rozptyl.

V praxi je žádoucí, aby byly výsledky konzistentní mezi dynamickým a statickým předpočítaným osvětlením. To znamená, že osvětlovací modely použité pro výpočet lightmap by měly být konzistentní s těmi, které se používají při dynamickém osvětlování v reálném čase. To pak umožňuje obě formy osvětlení vzájemně kombinovat bez nežádoucích vizuálních nesrovnalostí.

Poslední, avšak zcela zásadní výhodou je příznivost path tracingu vůči paralelizaci. Výpočet osvětlení uloženého do lightmap lze přirozeně definovat jako paralelní výpočetní problém, neboť každý texel lightmapy lze zpracovávat individuálně. Při samotném výpočtu se texely vzájemně neovlivňují, což umožňuje rozdělit práci mezi více nezávislých výpočetních jednotek a optimálně využít dostupných zdrojů.

2.1 Požadované funkce nástroje

Na základě předchozí analýzy a definovaných cílů jsem na svůj nástroj kladl následující požadavky. Nástroj musí umožňovat interaktivní vizualizaci scény s možností volného pohybu kamery. Scéna musí podporovat dva typy objektů – geometrický objekt (*mesh*) a světelný zdroj (*light*). Uživatel může objekty vybírat a upravovat jejich parametry v závislosti na jejich typu.

V případě světelných zdrojů má možnost jejich vytváření a mazání, změny umístění, orientace a další parametrů. S geometrickými objekty nelze přímo manipulovat, ale lze konfigurovat jejich vlastnosti související s generováním lightmap. Konkrétně jde o možnost vytvořit lightmapový atlas s uživatelem zvoleným rozlišením a následně pro tento atlas vygenerovat lightmapu.

Vzhledem k demonstrační povaze nástroje jsem se rozhodl podporovat různé formáty lightmap – jak nesměrové, tak směrové. Aby bylo možné snadno porovnávat různé výsledky, je možné pro každý geometrický objekt vytvořit více lightmapových atlasů s různým rozlišením a zároveň pro každý lightmapový atlas vygenerovat více lightmap různých formátů.

Možnost vygenerovat více lightmap pro jeden geometrický objekt lze vnímat i jako způsob simulace různých světelných podmínek. To je praktické například ve videohrách, kde se stejná scéna může vyskytovat za různých situací – například během jasného dne, při soumraku nebo v noci. Podobně lze tuto funkci využít i pro interiéry, kde lze předpočítat varianty osvětlení pro stav s rozsvícenými a zhasnutými světly.

Tato funkce však nebyla primárně navržena pro tyto účely a samotný nástroj neumožňuje definovat různé světelné konfigurace jinak než ruční úpravou scény – přepsáním původní konfigurace nebo vytvořením její kopie s odlišným nastavením světelných zdrojů.

Pro lepší interpretaci nástroj musí poskytovat pomocná vizualizační rozhraní, jako je náhled lightmapového atlasu a zobrazení hustoty texelů na odpovídajícím geometrickém objektu.

2.2 Reprezentace scény

Pro účely lightmappingu slouží scéna v první řadě jako vstupní data určující geometrii, její materiálové vlastnosti a rozmístění a vlastnosti světelných zdrojů. V druhé řadě pak slouží zároveň jako výstupní data, která scénu doplňují o vygenerované lightmapy a jejich lightmapový atlas.

Její struktura je založena na hierarchii uzlů, kde každý objekt může být umístěn ve scéně a být propojen s jinými objekty. Tyto objekty mohou nabývat tří základních typů:

- Prázdný objekt (*empty*)⁸ – slouží jako pomocný uzel pro správu hierarchie a transformací
- Geometrický objekt (*mesh*) – reprezentuje trojúhelníkovou síť
- Světlo (*light*) – světelný zdroj

Společnými vlastnostmi každého objektu jsou:

- Název objektu (unikátní v rámci scény)
- Rodičovský objekt – nadřazený objekt v hierarchii
- Transformace (pozice, rotace, měřítko) – definovaná relativně vůči rodičovskému objektu

2.2.1 Materiál

Jelikož data, která je třeba ukládat v rámci geometrie jsou ovlivněna rysy konkrétního materiálu, jímž je geometrie pokryta, je vhodné se nejprve seznámit s parametry materiálu a způsobem, jakým ovlivňuje reprezentaci povrchu.

Charakteristika každého materiálu je definována jeho BSDF a kombinací příslušných parametrů, které určují, jakým způsobem povrch interaguje se světlem.

V mém nástroji jsou všechny povrchy považovány za ideálně difúzní, které se řídí lambertovskou BRDF. To znamená, že odražené světlo se šíří do všech směrů rovnoměrně se stejnou intenzitou bez ohledu na směr, ze kterého je povrch pozorován. Oproti fyzikálně přesné formulaci lambertovské BRDF však můj nástroj neprovádí dělení albeda hodnotou π . To znamená, že pokud na osvětlovaný bod dopadá kolmo paprsek bílého světla s intenzitou 1, výsledná barva povrchu odpovídá přímo barvě jeho albeda. Tento přístup tak není založený na fyzikálních jednotkách, ale upřednostňuje jednoduchost a uživatelskou přívětivost.

Lambertovský model BRDF byl zvolen zejména proto, že je jednoduchý na implementaci a zároveň dostatečný pro demonstraci předpočítaného difúzního osvětlení. Zvolené BSDF tedy dokáže simulovat pouze difúzní složku, zatímco spekulární složka modelována není. Přestože je možné předpočítané osvětlení využít i pro pokročilejší BSDF se spekulární složkou, vyžaduje to složitější metody, které přesahují rámec této práce.

⁸ Prázdný objekt je pomocným objektem, který umožňuje provázání dvou a více objektů pod jeden společný rodičovský objekt. Každý objekt vytváří vlastní hierarchii, jejíž transformace se automaticky propaguje do pod-objektů. Díky tomu lze aplikovat změny polohy, rotace nebo měřítka na celý soubor propojených objektů prostřednictvím jediného nadřazeného uzlu.

S touto BSDF lze tedy věrně modelovat matné materiály jako je beton, dřevo či omítka. Nedokáže však správně modelovat lesklé materiály jako leštěné kameny, plasty, kovy či vlhké povrchy.

Nicméně kromě BRDF obsahuje zvolené BSDF také transmisní složku (BTDF), která umožňuje simulaci průhlednosti, resp. průsvitnosti materiálů. Průsvitnost je definována alfa kanálem barvy albeda, která určuje míru propustnosti světla skrz povrch materiálu.

Je však důležité zdůraznit, že tato BTDF nepopisuje úplnou interakci světla uvnitř objemu materiálu, ale pouze chování světla na rozhraní mezi materiálem a okolním prostředím. To znamená, že světlo při průchodu materiálem nemění svůj směr.

Tento jev je zásadní pro materiály jako je sklo, voda, plastové folie či závěsy. Důležitost tohoto jevu však spočívá i v tom, že průhledné textury se často využívají k simulaci detailních tvarů, které by jinak vyžadovaly ruční modelování pomocí 3D geometrie. To snižuje geometrickou složitost scény a tím optimalizuje výkon vykreslování.

Tento přístup se běžně používá například u listů stromů, trávy, drátěných plotů, vlasů nebo jemných ornamentálních prvků, kde by modelování každého detailu jako samostatné geometrie bylo neefektivní.

Daný materiál je pak pro účely mého nástroje definován následujícími parametry:

- Barva albeda (implicitně bílá) – určuje barvu povrchu, není-li definována textura albeda
- Textura albeda (nepovinný) – určuje barvu povrchu
- Textura normálové mapy (nepovinný) – upravuje povrchovou normálu při vykreslování
- Intenzita normálové mapy (nepovinný) – faktor ovlivňující míru intenzity normálové mapy

2.2.2 Geometrický objekt

Geometrický objekt představuje objekt ve scéně, jehož reprezentace popisuje povrch geometrie.

Zaměříme-li se na samotnou geometrii, ta se v případě grafických aplikací, jako jsou videohry, nachází ve své koncové podobě – tedy ve formě, která již nemusí odpovídat reprezentaci vhodné pro její produkci v modelovacím softwaru, ale je optimalizována pro vykreslování. Proto je běžné ji reprezentovat pomocí trojúhelníkových sítí, které jsou nativní reprezentací pro GPU.

Trojúhelníková síť je v nástroji reprezentována pomocí vrcholů a trojúhelníků, které jsou definovány indexy propojujících vrcholy do trojúhelníkových ploch. Vrcholy jsou uloženy relativně vůči transformaci objektu ve scéně.

Každý trojúhelník objektu je pokryt materiálem, který udává jeho povrchové vlastnosti. Konkrétní materiál pak udává, jaké atributy bude třeba ukládat společně s vrcholy sítě.

V případě materiálu v mém nástroji je každý vrchol definován následujícími atributy, které jsou nezbytné pro správné stínování a aplikaci textur:

- Pozice – určuje souřadnice vrcholu v prostoru
- Normála (normálový vektor) – definuje orientaci povrchu v daném bodě

- Tangenta (tečný vektor) – doplňuje normálový vektor a společně s ním umožňuje použití normálových map pro simulaci povrchových nerovností⁹
- Primární souřadnice UV mapování – určují, jak se textury albeda a normálové mapy promítají na povrch geometrie
- Sekundární souřadnice UV mapování – určují, jak se textura lightmapy promítá na povrch geometrie

Pro upřesnění je vhodné uvést, že ne všechny atributy se využívají ve všech případech. Například pokud není použita textura albeda nebo normálová mapa, primární kanál UV mapování zůstává nevyužitý. Podobně, pokud se na materiál neaplikuje lightmapa, nevyužívá se sekundární kanál UV mapování.

Pro zjednodušení implementace však nástroj používá jednotný formát vrcholu obsahující všechny uvedené atributy, přestože některé z nich nemusí obsahovat platná data. V praxi tedy využití jednotlivých atributů závisí na konkrétní konfiguraci materiálu a aplikovaných texturách.

2.2.3 Lightmapový atlas

Lightmapový atlas představuje unikátní parametrizaci trojúhelníkové sítě rozvinuté do 2D prostoru. Při parametrizaci dochází k rozdělení původní geometrie do menších segmentů – chartů, jejichž rozlišení (měřítko) a rozmístění v atlasu se řídí konfigurací zvolenou uživatelem při jejich vytváření. Výsledná parametrizace je tedy vždy uzpůsobena konkrétním rozměrům lightmapy.

To znamená, že atlas nelze snadno přizpůsobit jinému rozměr textury (lightmapy), aniž by bylo nutné přemístit charty uvnitř atlasu. Ačkoli UV souřadnice obecně nejsou vázány na konkrétní rozměry textury, v případě lightmapy je nutné zohlednit mezeru mezi charty, která zabraňuje vzorkování osvětlení ze sousedních chartů v důsledku bilineárního či bikubického filtrování.

Z tohoto důvodu není možné souřadnice UV mapování libovolně škálovat, neboť jsou optimalizovány pro konkrétní rozměry lightmapy. Při zmenšení atlasu by se mezera mezi charty mohla stát nedostatečnou, což by vedlo k nežádoucím artefaktům v podobě úniku světla napříč charty. Naopak zvětšení atlasu artefakty nezpůsobí, avšak vede k neefektivnímu využití dostupného texturového prostoru atlasu.

S ohledem na tyto faktory jsem se rozhodl nepodporovat přímou změnu velikosti atlasu, a místo toho umožnit vytváření více atlasů pro různé konfigurace rozlišení.

2.2.4 Lightmapa

Lightmapa je texturou, jejíž texely nesou údaje o osvětlení konkrétní části povrchu. To, jaké části povrchu daný texel připadá je dáno lightmapovým atlasem.

Lightmapy jsou svým způsobem podobné materiálům v tom, že rovněž pokrývají povrch geometrie. Intuitivně si je lze představit jako další vrstvu, která je aplikována na povrch geometrie společně s materiálem. Hlavním rozdílem oproti materiálu je, že reprezentujeme-li

⁹ Normálová mapa je definována vůči tečnému prostoru. Při aplikaci normálové mapy je zapotřebí zkonstruovat ortogonální bázi tečného prostoru, jež je definována pomocí tří vzájemně kolmých vektorů – normály, tangenty a bitangenty.

geometrii pomocí jednotlivých geometrických objektů, povrchy objektu mohou být pokryty několika různými materiály. V případě mého nástroje však lightmapa vždy pokrývá celý povrch objektu bez ohledu na materiál.

Ačkoli by bylo možné generovat více samostatných lightmap pro jednotlivé části objektu podle materiálů, v praxi by to bylo nepraktické, neboť by to vedlo ke zvýšení výpočetních nároků při vykreslování kvůli častějšímu přepínání textur.

V praxi existují i pokročilejší řešení, která se využívají při vykreslování více instancí stejného geometrického objektu různě umístěných ve scéně. V takovém případě je efektivnější nepoužívat jednu lightmapu na geometrický objekt, nýbrž využít sdílenou lightmapu, jež uchovává osvětlení pro všechny instance daného geometrického objektu. Nadstavbou tohoto řešení je vytvoření jedné globální lightmapy, která uchovává osvětlení pro celou scénu. (O'Donnell 2018, s. 98–99)

2.2.5 Světelné zdroje

Posledními prvky ve scéně jsou světelné zdroje. Ty jsem se rozhodl modelovat výhradně jako bezrozměrné, které ač nejsou fyzikálně přesné, jsou snadno implementovatelné. Navíc jsou blízké i herním enginům, v kterých se dodnes díky své výpočetní efektivitě běžně využívají. V kontextu předpočítaného osvětlení to znamená, že všechny stíny budou přirozeně ostré.

Mezi konkrétní podporované světelné zdroje jsem zařadil:

- Směrové světlo
- Bodové světlo
- Reflektorové světlo

Směrové světlo se v případě mé reprezentace scény řídí standardním vzorcem, který dopadající světlo modeluje jako nekonečně vzdálené světlo přicházející z určitého směru. Pro modelování bodového a reflektorového světla jsem však zvolil vzorců, použitých v herním enginu *Frostbite* (Lagarde a de Rousiers 2014, s. 28–33).

Útlum bodového světla¹⁰ se vzdáleností se tradičně řídí zákonem převrácených čtverců, tedy že intenzita světla klesá s druhou mocninou vzdálenosti od zdroje světla. Tento model je fyzikálně přesný, avšak přestože se s rostoucí vzdáleností blíží nule, nikdy skutečné nuly nedosáhne. Z hlediska optimalizace je však výhodné, pokud je možné některé světelné zdroje z výpočtu zcela vyřadit, protože již neovlivňují osvětlení daného bodu. V enginu *Frostbite* je proto použito odlišné definice, která dosah bodového světla ohraničuje určitou limitní vzdáleností od počátku zdroje. Tato upravená definice byla hlavním důvodem pro její volbu v mém nástroji.

Světelné zdroje jsou určeny následujícími parametry:

- **Typ / Type**
Světelný zdroj může být směrový, bodový nebo reflektorový.
- **Umístění / Position**
(*irrelevantní pro směrový zdroj*)

¹⁰ Reflektorový světelný zdroj je variantou bodového světelného zdroje, přičemž oblast vyzařovaného světla je omezena na kužel o určitém úhlu. Útlum světla se vzdáleností se tak vztahuje i na něj, přičemž vzdálenost se měří podél jeho osy od počátku směrem k podstavě.

Pozice počátku zdroje světla v prostoru.

- **Orientace / Rotation**

(irelevantní pro bodový zdroj)

Rotace světelného zdroje v prostoru.

- **Barva / Color**

Barva emitovaného světla.

- **Dosah / Range**

(irelevantní pro směrový zdroj)

Maximální vzdálenost od počátku zdroje, do které může světlo působit. Pro reflektorový zdroj odpovídá výšce kužele.

- **Intenzita / Power¹¹**

Určuje sílu emitovaného světla.

- **Úhel / Angle (0,0–180,0°)**

(relevantní pouze pro reflektorový zdroj)

Úhel vnějšího kužele reflektorového světla, pouze pro reflektorový zdroj. Určuje velikost podstavy kužele, resp. maximální oblast, ve které může světlo působit.

- **Přechod / Blend (0,0–1,0)¹²**

(relevantní pouze pro reflektorový zdroj)

Určuje měkkost přechodu mezi vnitřním a vnějším kuželem reflektorového zdroje.

- 0,0: znamená ostrý přechod.
- 1,0: znamená plynulý přechod.

Speciálním typem světelného zdroje je obloha (*ambientní světlo*), které představuje rozptýlené světlo oblohy. V tomto nástroji je implementováno jako jednoduchá, jednobarevná aproximace nepřímého osvětlení, která slouží k základnímu prosvětlení scén v oblastech, kam nedopadá přímé světlo.

2.3 Použité technologie

Vzhledem k požadavkům kladeným na nástroj, zejména podpoře interaktivní vizualizace scény, generování lightmap metodou path tracingu a serializaci/deserializaci dat, bylo nutné zvolit vhodný technologický stack.

Následující kapitola se věnuje přehledu použitých knihoven a obecně technologií, které byly zvoleny s ohledem na výkon, flexibilitu a snadnou integraci do navržené architektury nástroje.

Pro implementaci byl zvolen jazyk C++, jelikož kombinuje vysoký výkon s možností provádění optimalizací na nízké úrovni, a zároveň umožňuje vhodně organizovat kód díky podpoře objektově orientovaného návrhu.

Dalším důvodem volby C++ je široká podpora existujících knihoven. Vyvíjet všechny potřebné komponenty zcela od základu by nebylo praktické, neboť pro jednotlivé dílčí problémy existují vysoce optimalizovaná řešení, jež byla navržena s důrazem na efektivitu a širokou použitelnost.

¹¹ Intenzita světla v mém nástroji neodpovídá žádné fyzikální jednotce, ale její hodnota se 1:1 používá jako násobitel barvy, jejichž součinem je hodnota odpovídající radiometrické veličině zářivosti

¹² Parametrizace reflektorového zdroje je inspirována parametrizací z modelovacího softwaru Blender (Blender Foundation 2025)

Využití těchto specializovaných knihoven umožňuje soustředit se na samotnou problematiku generování lightmap, aniž by bylo nutné řešit implementaci a optimalizaci již dobře prozkoumaných oblastí.

V rámci vývoje nástroje byly v omezené míře využity i části externího zdrojového kódu, které byly upraveny a přizpůsobeny konkrétním potřebám nástroje. Tyto části však nejsou z hlediska odborného obsahu této práce zásadní, a proto nejsou v textu podrobně rozebírány. Veškeré takto převzaté části jsou v komentářích zdrojového kódu řádně označeny včetně odkazu na původní zdroj.

Přestože byl nástroj vyvíjen primárně pro operační systém Windows, všechny použité technologie a knihovny jsou přenositelné i na jiné platformy, bude-li nástroj kompilován pro odpovídající cílové prostředí. V případě knihoven distribuovaných v binární podobě je pak nutné použít verze zkompilevané pro danou platformu.

A) Grafické API a vizualizace

- **OpenGL 4.6** (The Khronos Group nedatováno c) – Používá se pro hardwarově akcelerované vykreslování 2D/3D geometrie a provádění výpočtů pomocí GPU. V kontextu nástroje slouží zejména pro vizualizaci scény a vykreslování uživatelského rozhraní. Mimo to jej však nástroj využívá i pro rasterizaci geometrie do texturového atlasu (viz 4.2.1)
- **GLAD** (Herberth nedatováno) – Stará se o generování loaderu pro dynamické načítání funkcí OpenGL odpovídajících zvolené verzi OpenGL API.
- **GLFW** (GLFW nedatováno) – Slouží pro vytváření oken, správu vstupů z klávesnice/myši a inicializaci OpenGL kontextu.
- **Dear ImGui** (Cornut nedatováno) – Poskytuje systém pro tvorbu uživatelského rozhraní na bázi paradigmatu *Immediate Mode GUI*.

B) Matematika

- **GLM (OpenGL Mathematics)** (G-Truc Creation nedatováno) – Matematická knihovna pro lineární algebru, využívaná pro vektorové a maticové operace.

C) Lightmapping a path tracing

- **xatlas** (Young nedatováno) – Umožňuje vytvářet optimální unikátní UV parametrizaci geometrie a jejich vhodné uspořádání do lightmapového atlasu dle zvolené konfigurace (viz 3).
- **Intel Embree** (Intel nedatováno a) – Zajišťuje vysoce výkonnou akceleraci výpočtů průsečíků paprsků s geometrií, což jednou ze základních úloh při path tracingu (viz 4.1.1 a 4.3.2).
- **Intel Open Image Denoise** (Intel nedatováno c) – Implementuje denoisingový filtr založený na strojovém učení k odstranění šumu z obrazů generovaných pomocí ray tracingu – v případě tohoto nástroje lightmap (viz 4.4).
- **Intel oneAPI Threading Building Blocks** (Intel nedatováno b) – Závislost knihoven Intel Embree a Intel Open Image Denoise, které jsou v rámci projektu použity.

D) Načítání/ukládání dat

- **stb_image (součást knihovny stb)** (Barrett nedatováno) – Používá se k načítání obrazových souborů (např. textur materiálů nebo lightmap) ve formátu PNG, JPG a dalších.
- **fastgltf** (Apeler nedatováno) – Zajišťuje načítání souborového formátu glTF, který se používá k reprezentaci 3D modelů (viz 5).
- **JSON for Modern C++** (Lohmann nedatováno) – Používá se pro serializaci a deserializaci formátu JSON (viz 5).

E) Pomocné knihovny

- **spdlog** (Melman nedatováno) – Výkonný a flexibilní systém logování pro výpis diagnostických informací při běhu aplikace.
- **stduuid** (Bancila nedatováno) – Používá se pro generování UUID (universally unique identifiers), která se používá pro jednoznačnou identifikaci lightmapových atlasů a lightmap (viz 5).
- **Native File Dialog Extended** (Teo nedatováno) – Poskytuje nativní dialogová okna pro otevírání a ukládání souborů na různých operačních systémech.

3 Tvorba lightmapového atlasu

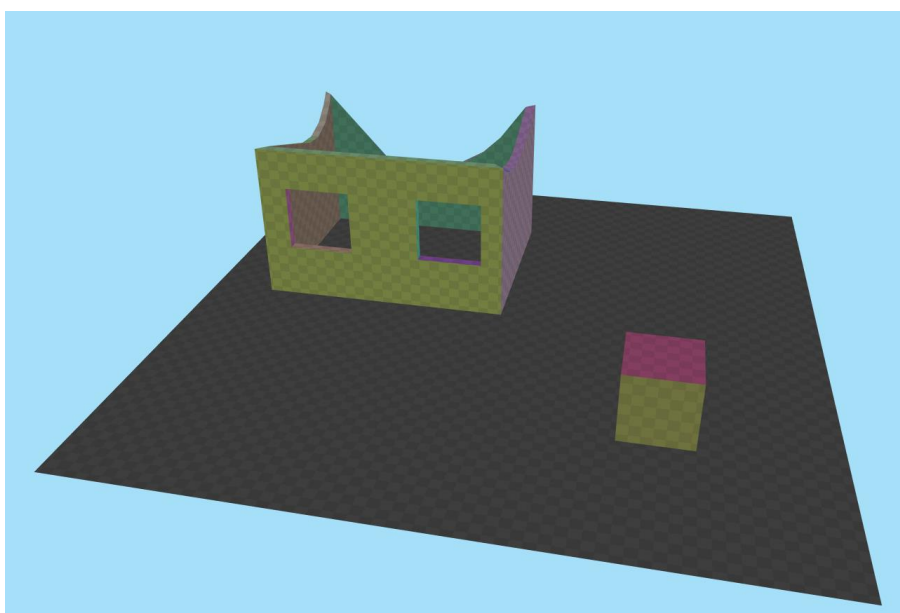
Samotnému procesu generování lightmap předchází vytvoření lightmapového atlasu, který definuje texturový prostor, v němž lze uchovávat předpočítané osvětlení pro povrch daného geometrického objektu. V mém nástroji se atlas vždy vytváří pouze pro jeden konkrétní geometrický objekt a neumožňuje tak, aby více objektů sdílelo společný atlas.

Při tvorbě atlasu dojde k rozvinutí – unikátní parametrizaci trojúhelníkové sítě do 2D prostoru textury o určitém rozměru, jenž určuje hranice texturového prostoru. Rozvinutím je každému trojúhelníku přidělena určitá jedinečná oblast texelů, v kterých lze uchovávat osvětlení.

Přestože lightmapový atlas je vázán na konkrétní velikost texturového prostoru, nejedná se o texturu jako takovou. Parametrizací dochází ke vzniku sekundárních souřadnic UV mapování, které slouží jednak pro jednoznačné přidělení oblasti texelů v texturovém prostoru a jednak pro pozdější mapování lightmapy na povrch geometrie.

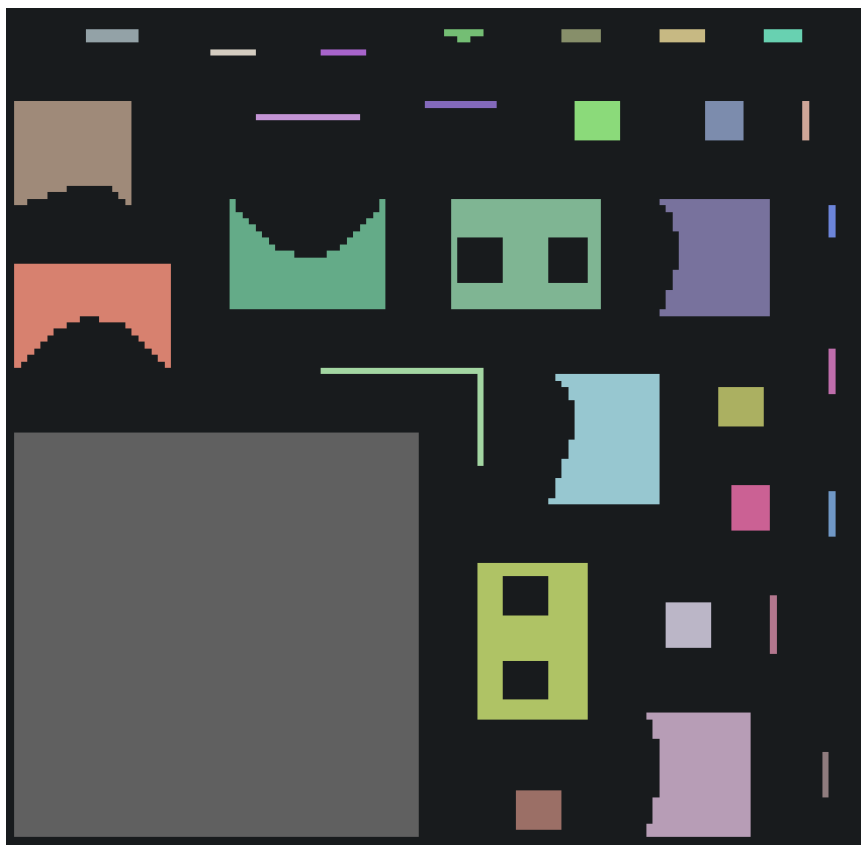
Samotná lightmapa pak musí být generována s ohledem na tento atlas, neboť ten předepisuje, který texel náleží jaké části povrchu.

Při parametrizaci trojúhelníkové sítě dochází k její segmentaci na tzv. charty. Ty obvykle odpovídají souvislým částem povrchu, které mají podobnou orientaci v 3D prostoru. Při tomto procesu je povrch rozvinut, resp. promítnut, do 2D prostoru, přičemž cílem je minimalizovat jeho deformace, jako je zkosení či roztažení, aby výsledné UV mapování co nejlépe odpovídalo původnímu tvaru geometrie.



Obr. 14: Povrch geometrie po rozvinutí do lightmapového atlasu – každý povrch je obarven podle příslušnosti ke konkrétnímu chartu v lightmapovém atlasu (viz Obr. 15). Políčka šachovnicového vzoru na površích geometrie odpovídají velikosti jednoho texelu v lightmapovém atlasu.

Zdroj: vlastní zpracování



Obr. 15: Povrch geometrie parametrizovaný a uspořádaný do lightmapového atlasu. Jednotlivé charty jsou barevně odlišeny.

Zdroj: vlastní zpracování

Segmentací dojde k oddělení některých částí povrchu v UV prostoru, které jsou jinak v 3D prostoru spojeny. V důsledku toho vznikají hrany, jež jsou sice v 3D prostoru sdílené, ale v UV prostoru se nacházejí na dvou odlišných místech. Proto vrcholy každé takové hrany musejí být rozdvojeny, aby mohly nést svoji vlastní sekundární souřadnici UV mapování, přičemž ostatní atributy vrcholu zůstávají pro obě instance stejné.

Vznik nových instancí vrcholů implikuje nutnost vytvoření nové trojúhelníkové sítě. Ta je odvozena z té původní, avšak je tvořena i nově vzniklými vrcholy se sekundárními souřadnicemi UV mapování a odpovídajícími trojúhelníky, jejichž indexy se na tyto vrcholy odkazují.

Při následném uspořádání chartů do atlasu je třeba zohlednit i způsob filtrování, jaký bude využit při vzorkování osvětlení z lightmapy. Při lightmappingu se typicky používá bilineárního či bikubického filtru, které vzorkují osvětlení z vícero sousedních texelů, přičemž výsledná hodnota je dána jejich interpolací. Aby se zabránilo úniku světla mezi sousedícími charty, musí mezi nimi být ponechána dostatečná mezera, která odpovídá rozsahu vzorkování daného použitým filtrem.

V mém nástroji je možné volit mezi bilineárním a bikubickým filtrováním, přičemž tato volba není vázána na konkrétní atlas, ale na globální nastavení nástroje, které ovlivňuje způsob filtrování lightmap napříč celou scénou. To znamená, že při vytváření atlasu se vychází z nejhoršího možného případu, kterým je bikubické filtrování, jež vzorkuje osvětlení z oblasti 4x4 texelů, tj. ± 2 od středu vzorkované oblasti v každém směru.

Vzhledem k tomu, že tvorba UV parametrizace představuje složitý optimalizační problém, byl pro její automatizované generování použit *xatlas* (Young nedatováno). Tato knihovna provádí unikátní UV parametrizaci trojúhelníkové sítě s cílem minimalizovat deformace a optimalizovat rozložení chartů v texturovém prostoru. Xatlas zároveň řeší rozložení chartů v lightmapovém atlasu tak, aby bylo co nejefektivnější a zároveň zajišťovalo dostatečné rozestupy mezi charty, které eliminují artefakty způsobené filtrováním textur.

API knihovny *xatlas* je v nástroji používáno tak, že nejdříve je vytvořena instance třídy „Atlas“, jež představuje kontext pro práci s knihovnou. Tento objekt slouží jako ústřední entita, která spravuje proces parametrizace geometrie a uspořádání chartů do výsledného atlasu.

Nejprve je do kontextu nahrán povrch geometrie ve formátu, jenž vyžaduje knihovna. Jde o vesměs totožnou reprezentaci trojúhelníkové sítě, jaká se používá i v rámci nástroje, jejíž vrcholy jsou však omezeny pouze na atributy pozice, normály a primárních souřadnic UV mapování. Souřadnice UV mapování nejsou knihovnou povinně vyžadovány, ale jsou-li zadány, lze z nich při parametrizaci vycházet.

Po vytvoření interní reprezentace geometrie pro *xatlas* je možné přistoupit k samotné konstrukci atlasu. Prvním krokem je segmentace povrchu geometrie do chartů. Tento proces lze do jisté míry konfigurovat nastavením parametrů, jako je například maximální povolená odchylka normál trojúhelníků od průměrné normály celého chartu, jejichž úpravou lze ladit možný práh deformací více do detailu. Avšak vzhledem k tomu, že nástroj na problematiku deformací neklade žádné konkrétní nároky a zároveň pro uživatele by konfigurace těchto parametrů mohla být příliš komplikovaná, rozhodl jsem se zvolit výchozí parametry poskytované knihovnou.

Po segmentaci následuje volání funkce pro uspořádání chartů do texturového prostoru s možností konfigurace. Zásadními jsou parametry pro volbu rozlišení a volbu velikosti mezery mezi jednotlivými charty.

Pro zvolení rozlišení atlasu nabízí knihovna dvě možnosti. První možností je zadání požadovaného rozměru atlasu v texelech, který je společný pro horizontální i vertikální rozměr. Druhou možností je zadání požadované hustoty texelů vůči jednotce délky ve 3D prostoru – tzv. *texels per unit*, tj. kolik texelů připadne na jednotkovou vzdálenost ve scéně, přičemž rozměr atlasu je dopočítán automaticky na základě zadané hustoty texelů.

Je však důležité upozornit, že i při explicitním zadání rozměru atlasu se tato hodnota chápe spíše jako vodítko. Knihovna se snaží tomuto rozměru přiblížit, ale výsledné rozměry se mohou mírně lišit – být větší či menší než původně zadaná hodnota.

V rámci nástroje jsou podporovány obě varianty zadávání rozlišení s tím, že hustota texelů je ve výchozím stavu rovna hodnotě 0. Hustota texelů se tedy použije pouze tehdy, je-li nastavena na nenulovou hodnotu a v opačném případě se používá rozměr textury.

Dalším parametrem konfigurovatelným na úrovni knihovny je velikost mezery v texelech, jež je ponechána na okraji každého chartu. Vzhledem k předpokladu použití bikubického filtrování při vzorkování lightmap, je velikost mezery vždy na nastavena na tři texely, které jsou dostatečné pro eliminaci nežádoucího úniku světla ze sousedních chartů i s určitou rezervou jednoho texelu, kterého je zapotřebí v důsledku metody použité pro rasterizaci atlasu do textury za účelem vytváření vzorků pro path tracing (viz 4.2.1).

Po dokončení tvorby atlasu je jeho obsah převeden do interní reprezentace, kterou nástroj využívá pro další zpracování. Jak již bylo zmíněno, při vytváření unikátní UV parametrizace dochází ke vzniku nových vrcholů, což si žádá vytvoření nové trojúhelníkové sítě. Nové vrcholy jsou však pouze kopiemi původních vrcholů s rozdílnými sekundárními souřadnicemi UV mapování pro každou z instancí původního vrcholu.

Xatlas je v tomto ohledu flexibilní, neboť jím vygenerovaná reprezentace atlasu poskytuje informaci o tom, z jakého původního vrcholu byl nový vrchol vytvořen a jaká je jeho sekundární souřadnice UV mapování. Díky tomu není nutné vytvářet novou trojúhelníkovou síť ihned, ale je možné uložit si pouze její kompaktní formu v této reprezentaci a vytvořit kompletní trojúhelníkovou síť až tehdy, kdy je to třeba.

Nástroj proto při ukládání atlasu do souboru využívá právě tuto zjednodušenou reprezentaci, čímž snižuje paměťovou náročnost. Skutečná trojúhelníková síť se rekonstruuje až při načtení dat ze souboru a následně se drží v paměti při běhu aplikace.

Pro účely pomocných vizualizací v nástroji je navíc zaznamenána i informace o tom, do kterých chartů byly jednotlivé trojúhelníky segmentovány. Díky tomu lze v nástroji vizuálně zvýraznit jednotlivé charty na původní geometrii například barevným odlišením (viz 6.1). Tato informace však slouží čistě k vizualizačním účelům a v kontextu lightmappingu nemusí být potřebná.

Posledním důležitým krokem při vytváření interní reprezentace atlasu je zaznamenání skutečného rozměru atlasu. Výsledný atlas vytvořený knihovnou xatlas může nabývat různých rozměrů. Obecně však platí, že z hlediska efektivity na úrovni GPU by textury měly mít rozměry odpovídající mocninám dvou (např. 256, 512, 1024 apod.). Textury s těmito rozměry jsou totiž optimálně zarovnány vůči paměťovým blokům a často těží z vyšší efektivity při jejich filtrování a dalších operacích prováděných grafickým hardwarem. Z tohoto důvodu je vhodné výsledný rozměr atlasu zaokrouhlit na nejbližší vyšší mocninu dvou.

V mém nástroji je proto skutečný rozměr atlasu po jeho vygenerování upraven na nejbližší vyšší mocninu dvou a zaznamenán jako součást interní reprezentace. To znamená, že rozměr atlasu se zvětší. Pouhé zvětšení atlasu při zachování původního rozlišení by však vedlo k neoptimálnímu využití celkového dostupného prostoru. Proto je navíc původní rozlišení přizpůsobeno novému rozměru atlasu tak, aby větší z rozměrů původní velikosti atlasu odpovídal nové velikosti atlasu. Aby byl zachován původní poměr stran a nedošlo k roztažení atlasu, musí být oba rozměry zvětšovány ve stejném měřítku. V případě, že atlas není čtvercový, tak při tomto přepočtu zůstává v druhém – menším – rozměru určitý prostor nevyužit.

Zvýšením rozlišení atlasu však dochází i k mírnému zvětšení původně zvolené mezery mezi charty, což může vést k méně optimálnímu využití texturového prostoru. Tento dopad je však v kontextu lightmappingu zanedbatelný, protože primárním cílem úpravy rozměru je dosažení optimální kompatibility s GPU při zachování správné interpolace texelů při filtrování.

4 Generování lightmap

S existujícím lightmapovým atlasem, definujícím texturový prostor pro uchovávání předpočítaného osvětlení pro povrch daného geometrického objektu, lze přistoupit k samotnému generování lightmap.

Generování lightmap v nástroji lze rozdělit do tří hlavních fází. První z nich je fáze *preprocessingu*, jejímž cílem je připravit veškerá podpůrná data, která budou následně využita při výpočtu osvětlení. V této fázi dochází například ke konstrukci interní reprezentace scény vhodné pro sledování paprsků, detekci texelů a jejich mapování na konkrétní místa na povrchu cílového geometrického objektu, kde bude osvětlení počítáno. Zároveň jsou zaznamenány i texely v mezerách mezi charty, které bude následně třeba vyplnit osvětlením pomocí tzv. dilatace, aby se předešlo artefaktům vznikajícím při filtrování lightmapy při vykreslování.

Druhou fází je samotný výpočet osvětlení, který využívá metody path tracingu. Nástroj byl navržen s ohledem na progresivní výpočet, jenž uživateli umožňuje sledovat průběžné výsledky po přírůstcích, které se průběžně zpřesňují. Pokud je dosažená kvalita výsledku pro uživatele dostačující, může výpočet kdykoliv předčasně ukončit a ponechat si aktuální výstup bez nutnosti čekat na úplné dokončení výpočtu. Tento přístup zvyšuje flexibilitu tím, že kompromis mezi kvalitou výsledku a časovou náročností zůstává plně v rukou uživatele.

Třetí fází je *postprocessing*, který následuje po výpočtu osvětlení. V této fázi se provádí dvě klíčové operace: výplň texelů v mezerách mezi charty (dilatace) a aplikace denoisingového filtru, který potlačuje šum vznikající vlivem Monte Carlo integrování. Jelikož jsou průběžné výsledky zobrazovány po přírůstcích, je třeba vyjasnit, co znamená „po výpočtu“ – postprocessing lze totiž provádět jak po úplném (nebo předčasném) ukončení výpočtu, tak i mezi jednotlivými přírůstky. Uživateli je proto navíc dána možnost provádět postprocessing i před zobrazením průběžného výsledku, což mu dává možnost získat lepší představu o skutečné konečné podobě lightmapy. To je ovšem vykoupeno nižší frekvencí, při jaké jsou průběžné výsledky zobrazovány. Tuto možnost lze kdykoliv za běhu aktivovat či deaktivovat. Dílčí kroky postprocessingu lze navíc zapínat jednotlivě. Například dilatace, která není výpočetně náročná, je ve výchozím stavu aktivní a provádí se při každém zobrazení mezivýsledku, zatímco náročnější denoising zůstává ve výchozím stavu deaktivován.

4.1 Akcelerační struktury pro výpočet osvětlení

Při návrhu této interní reprezentace bylo důležité zajistit, aby obsahovala veškeré informace potřebné pro výpočet osvětlení, a zároveň umožňovala efektivní a snadný přístup k těmto datům během jednotlivých operací výpočtu. Za tímto účelem byly zvoleny vhodné akcelerační struktury, které reflektují specifika daného výpočetního problému.

Základním předpokladem při generování lightmapy je, že scéna je považována za zcela statickou. V kontextu nástroje to platí zejména pro osvětlení, neboť s geometrickými objekty nelze manipulovat. Dojde-li však poté k jakékoliv změně světelné konfigurace – například k úpravě parametrů některého ze světelných zdrojů – osvětlení uložené v lightmapě přestane odpovídat aktuálnímu stavu scény. V takovém případě je nutné lightmapu znovu přepočítat, aby reflektovala nové světelné podmínky.

V momentě, kdy je zahájeno generování lightmapy, je vytvořena dočasná interní kopie současného stavu scény ve formě reprezentace, která je vhodná pro sledování paprsků. Tato reprezentace slouží výhradně pro účely výpočtu osvětlení pro konkrétní lightmapu a není dále ovlivňována následnými změnami ve scéně. Tento přístup zajišťuje konzistenci výpočtu – například v situaci, kdy by uživatel během běhu výpočtu upravil parametry světla, by bez takové izolace mohlo dojít k narušení determinismu a vzniku nekonzistentních nebo chybných výsledků.

4.1.1 Akcelerační struktury pro geometrii

Nejprve se zaměříme na reprezentaci geometrie scény. Pro její potřeby je klíčové být schopen se efektivně dotazovat na průnik daného paprsku s povrchem geometrie a v případě nalezeného průsečíku určit odpovídající povrchové atributy (např. normálu, albedo a jiné). Tyto dvě úlohy lze chápat jako oddělené: nejprve je třeba průsečík nalézt, teprve poté je možné získat informace o materiálu a geometrii v daném bodě.

Pro efektivní hledání průsečíků paprsku s povrchem bylo zvoleno akcelerační struktury zprostředkované knihovnou *Intel Embree* (Intel nedatováno a), jež nabízí vysoce optimalizované algoritmy navržené pro tento typ úloh. Její vytvoření je přímočaré a spočívá v nahrání pozic vrcholů a trojúhelníků s indexy vrcholů do reprezentace scény, používané knihovnou.

Při jejím vytváření jsou vytvořeny dva globální buffery – jeden obsahuje pozice všech vrcholů ve scéně a druhý trojúhelníky definované jako trojice indexů do tohoto bufferu vrcholů. Vzhledem k tomu, že jde o globální buffery sdílené napříč celou scénou, není možné, aby pozice vrcholů zůstaly definovány relativně vůči jednotlivým geometrickým objektům. Proto jsou všechny pozice vrcholů před nahráním do bufferů transformovány do světového prostoru.

Pro účely nalezení průsečíku je taková reprezentace zcela dostačující. Při dotazu na průsečík paprsku s geometrií dokáže knihovna *Intel Embree* vrátit buď informaci o tom, že k žádnému zásahu nedošlo, nebo podrobnosti o zasaženém povrchu. Konkrétně se jedná o index zasaženého primitiva (trojúhelníku) společně s barycentrickými souřadnicemi, jež určují přesné místo zásahu na jeho ploše.

Pro vyhodnocení interakce paprsku světla s povrchem však samotné geometrické informace nestačí – je nutné znát i materiálové vlastnosti zasaženého místa. Proto si nástroj vedle akcelerační struktury knihovny *Intel Embree* udržuje obdobně i vlastní globální buffery s dalšími atributy vrcholů – jmenovitě normálami, tangentami a primárními souřadnicemi UV mapování.

Jestliže však *Intel Embree* vrací index zasaženého trojúhelníku, je třeba rovněž znát, jaký materiál danému trojúhelníku přísluší. Toho je docíleno globálním bufferem, který uchovává indexy materiálů pro každý trojúhelník ve scéně. Samotné materiály, včetně textur jsou odkazovány přímo ze vstupní scény, neboť ty se po jejím načtení v případě mého nástroje měnit nemohou, a proto je není třeba duplikovat.

Při vytváření těchto bufferů je zvláštní pozornost věnována cílovému objektu, tj. geometrickému objektu, na jehož povrch se bude generovaná lightmapa aplikovat. Pro jeho potřeby vzniká samostatná datová struktura, reprezentující tzv. prostor lightmapy – tedy 2D texturový prostor odpovídající parametrizaci cílového objektu. Tato reprezentace v sobě uchovává předpočítaná data pro akceleraci výpočtu barycentrických souřadnic uvnitř 2D trojúhelníků, jejichž vrcholy jsou dány souřadnicemi sekundárního UV mapování cílového objektu. Zároveň jsou

zaznamenány i offsety do globálních bufferů, aby se vědělo, v které části bufferu začíná geometrie cílového objektu.

Tato struktura slouží jako akcelerační prostředek pro efektivní výpočet barycentrických souřadnic 2D bodu vůči trojúhelníku v prostoru lightmapy. Jinými slovy, umožňuje rychleji určit, zda a kde se daný 2D bod nachází uvnitř určitého trojúhelníku na parametrizovaném povrchu cílového objektu.

Při předpočítání každého trojúhelníku se vypočítají vektory směřující z referenčního vrcholu („počátku“) do zbývajících dvou vrcholů, a zároveň se spočítá převrácená hodnota obsahu rovnoběžníku¹³ pro následný výpočet barycentrických souřadnic.

Výsledná struktura tak obsahuje všechna data potřebná k tomu, aby šlo souřadnice libovolného bodu v rámci 2D parametrického prostoru převést na barycentrické souřadnice vůči danému trojúhelníku pomocí méně operací, než kdyby se měla tato data počítat opakovaně při každém dotazu na barycentrické souřadnice. Tyto barycentrické souřadnice jsou pak dále využívány pro interpolaci atributů uvnitř povrchu trojúhelníku, nebo pro zjištění, zda se daný bod skutečně nachází uvnitř daného trojúhelníku.

```
struct Triangle2dBarycentricData
{
    glm::vec2 vec_ab;
    glm::vec2 vec_ac;
    glm::vec2 tri_a;
    float inv_denom;

    constexpr auto get_barycentric_coordinates_uv(const glm::vec2& point) const
    -> glm::vec2
    {
        glm::vec2 vec_ap = point - tri_a;

        // Compute the signed area of triangle ACP divided by signed area of triangle ABC
        float u = (vec_ap.x * vec_ac.y - vec_ac.x * vec_ap.y) * inv_denom;

        // Compute the signed area of triangle ABP divided by signed area of triangle ABC
        float v = (vec_ab.x * vec_ap.y - vec_ap.x * vec_ab.y) * inv_denom;

        return {u, v};
    }

    constexpr static auto precompute(const glm::vec2& tri_a, const glm::vec2& tri_b,
                                     const glm::vec2& tri_c) -> Triangle2dBarycentricData
    {
        glm::vec2 vec_ab = tri_b - tri_a;
        glm::vec2 vec_ac = tri_c - tri_a;

        // Compute the denominator, i.e. signed area of triangle ABC
        // NOTE: The actual area is half this value, but since we'll divide other areas by
        // the same expression, the 0.5 factor cancels out in all barycentric coordinate
        // calculations.
        float denominator = vec_ab.x * vec_ac.y - vec_ac.x * vec_ab.y;

        return Triangle2dBarycentricData{.vec_ab = vec_ab,
                                          .vec_ac = vec_ac,
                                          .tri_a = tri_a,
                                          .inv_denom = 1.0f / denominator};
    };
};
```

Obr. 16: Zdrojový kód akcelerační struktury pro efektivnější výpočet barycentrických souřadnic ve 2D prostoru

Zdroj: vlastní zpracování

¹³ Mějme trojúhelník ABC a pod P, jehož barycentrické souřadnice chceme určit, potom lze tyto souřadnice vyjádřit jako poměr obsahů dílčích trojúhelníků ACP, ABP a BCP vůči obsahu trojúhelníku ABC. Jestliže se obsah trojúhelníků počítá jako $\frac{1}{2} \cdot \text{strana} \cdot \text{výška}$, zůstávají poměry mezi obsahy trojúhelníků zachovány i při vypuštění faktoru $\frac{1}{2}$, což odpovídá obsahu rovnoběžníku.

4.1.2 Akcelerační struktury pro světelné zdroje

Mimo geometrie musí interní reprezentace scény pro sledování paprsků zahrnovat i světelné zdroje. I v jejich případě lze provádět optimalizace, jež umožňují osvětlení vyhodnocovat s vyšší efektivitou.

Pro každý světelný zdroj ve scéně je vytvořena jeho interní kopie v podobě datové struktury, jež je optimalizovaná pro účely výpočtu osvětlení. V praxi to znamená, že některé parametry určené primárně pro uživatelské zadávání – například orientace světla pomocí rotace či úhel a přechod reflektorového světla – jsou převedeny do výpočetně efektivnější reprezentace, která usnadňuje a urychluje výpočty při path tracingu. V opačném případě by tyto parametry musely být počítány opakovaně, což by negativně ovlivňovalo výpočetní režii.

Mimo těchto základních optimalizací na úrovni datové reprezentace lze přistoupit i k prostorovému omezování jejich vlivu. Při návrhu světelných zdrojů jsem zvolil takové modely bodového a reflektorového světla, jejichž působnost je omezena pouze na určitý objem – kouli o určitém poloměru v případě bodového světla a kužel o určité výšce a poloměru podstavy v případě reflektorového světla. Tyto tvarové charakteristiky lze odvodit z parametrů daného světelného zdroje.

Díky tomu, že každé světlo působí jen v omezené oblasti, lze pro libovolný bod ve scéně předem určit, zda na něj dané světlo může mít vliv. Za tímto účelem je v nástroji implementována tzv. *light grid* – jednoduchá jednoúrovňová pravidelná 3D mřížka tvaru kváдру, která pokrývá celý prostor scény. Každá buňka mřížky pak obsahuje seznam indexů světelných zdrojů, které potenciálně působí na libovolný bod nacházející se uvnitř dané buňky. Počet buněk mřížky v každém směru 3D prostoru scény je uživatelsky konfigurovatelný.

Tento přístup výrazně přispívá k optimalizaci výpočtu osvětlení, neboť umožňuje omezit množinu světelných zdrojů, které je třeba při výpočtu osvětlení daného bodu brát v úvahu. Namísto všech světél ve scéně je možné pracovat pouze s podmnožinou světél přiřazených k buňce, do které daný bod spadá. Tento efekt se nejvýrazněji projevuje ve scénách s větším množstvím světelných zdrojů, kde pomáhá redukovat výpočetní režii při lokálním osvětlování bodu.

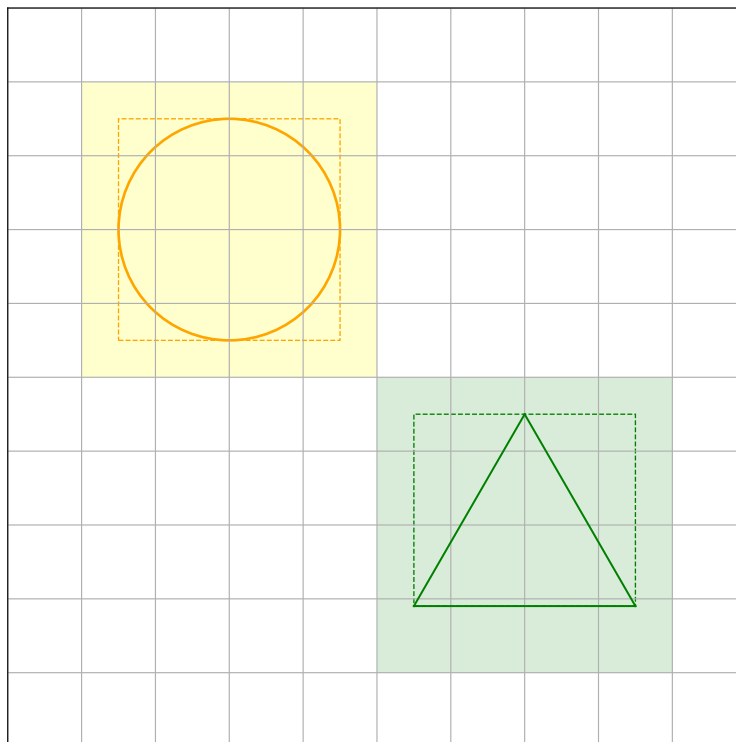
Skutečná efektivita tohoto řešení však závisí na velikosti oblasti působení jednotlivých světelných zdrojů, jejich rozmístění ve scéně a také na zvolené hustotě mřížky. Pokud je v jediné buňce koncentrováno příliš mnoho světelných zdrojů, nemusí být výpočetní úspora tolik výrazná. Tento problém by bylo možné zmírnit například použitím víceúrovňové adaptivní mřížky s proměnlivou velikostí buněk, které by se automaticky přizpůsobovaly lokální hustotě světelných zdrojů. Ve složitějších oblastech scény s vyšší koncentrací světél, by se mřížka zjemnila, a naopak v méně exponovaných oblastech by buňky mohly zůstat větší.

Navzdory těmto omezením představuje použití *light grid* za běžných okolností účinný a implementačně nenáročný způsob prostorového třídění světél pro potřeby výpočtu osvětlení.

Při konstrukci mřížky je nejprve nutné určit, které světelné zdroje ovlivňují jakou část prostoru. Přiřazení světelného zdroje k jednotlivým buňkám mřížky je určeno na základě průniku jeho ohraničujícího objemu s danou buňkou. Každý světelný zdroj je při této proceduře reprezentován pomocí svého *axis-aligned bounding boxu* (AABB) – tedy kváдру, zarovnaného

s osami souřadného systému, který ohraničuje objem působnosti světelného zdroje. Pokud AABB světelného zdroje protíná danou buňku mřížky, je daný zdroj do této buňky zaregistrován.

Specifickým případem je směrový světelný zdroj, jenž je modelován jako nekonečně vzdálené světlo působící v určitém směru. Na rozdíl od ostatních typů nemá omezenou oblast působení a potenciálně tak ovlivňuje každý bod ve scéně. Z tohoto důvodu nemá žádný ohraničující objem, resp. jsou jím hranice scény. Tato skutečnost implikuje přiřazení všech směrových světel každé buňce mřížky. Aby se zamezilo neefektivnímu využití paměti, nejsou směrové zdroje světla zahrnuty do indexů uvnitř buněk, ale existuje pro ně jeden globální seznam, který uchovává indexy zdrojů světla, jež působí na každý bod uvnitř mřížky.



Obr. 17: Zjednodušené schéma 2D průřezu light grid zobrazující přiřazení světelných zdrojů jednotlivým buňkám mřížky – bodový zdroj (žlutě) a reflektorový zdroj (zeleně)

Zdroj: vlastní zpracování

4.2 Preprocessing

Ještě před samotným výpočtem osvětlení metodou path tracingu je nezbytné provést určité přípravné kroky vytvářející podpurná data, jež zajišťují optimální průběh výpočtu osvětlení. Do fáze preprocessingu patří konstrukce již představených akceleračních struktur, ale spadá sem rovněž i detekce texelů a jejich mapování na konkrétní místa na povrchu cílového geometrického objektu, kde bude osvětlení počítáno. To zahrnuje tvorbu vzorků na povrchu cílového objektu, které určují pozice, z nichž budou do scény vrženy primární paprsky. Současně se generuje datová struktura označovaná jako dilatační mapa, jež slouží k pozdější výplni texelů v mezerách mezi charty, aby se předešlo artefaktům při filtrování lightmapy.

4.2.1 Vytváření vzorků pro primární paprsky

První úlohou při realizaci path tracingu je získání pozic paprsků, odkud bude iniciována simulace šíření světla ve scéně. V tomto okamžiku již máme k dispozici lightmapový atlas, jenž definuje texturový prostor pro daný geometrický objekt, a také reprezentaci scény uzpůsobenou pro sledování paprsků.

Díky atlasu víme, jakým způsobem je povrch geometrie rozvinut do 2D texturového prostoru. Tato informace nám umožňuje určit, které části povrchu jednotlivé texely lightmapy pokrývají. Na základě toho je možné pro každý texel vytvořit reprezentativní množství vzorků – tedy bodů, které leží uvnitř daného texelu a odpovídají konkrétním místům na povrchu ve 3D prostoru.

Mimo samotné pozice vzorku je ale zároveň třeba určit i další geometrické atributy, které se vážou k danému místu na povrchu. Konkrétně jde o normálu a tangentu, jež lze v místě vzorku získat interpolací vrcholových atributů trojúhelníku, na jehož ploše se vzorek nachází.

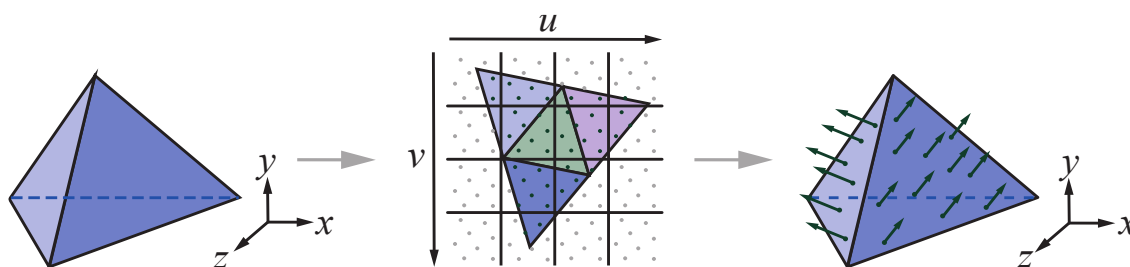
V tomto kontextu je oblastí vzorkování plocha texelu, přesněji řečeno její část, která se překrývá s povrchem geometrie. Výstupem jsou body, z kterých lze vrhat primární paprsky do scény, přičemž výsledné osvětlení získané po jejich průchodu scénou je akumulováno zpět do vzorkovaného texelu. Pokud do texelu žádný trojúhelník nezasahuje, není třeba pro takový texel žádné osvětlení počítat.

Samotný výpočet osvětlení je však další oblastí vzorkování, která pro stejný bod primárního paprsku může vrátit odlišnou hodnotu v závislosti na průchodu paprsku scénou. Pro zachycení nepřímého osvětlení si tak lze vystačit i s jediným bodem vzorkovaným na ploše texelu.

Přímočarým řešením je generovat vzorky ve středech jednotlivých texelů. Takové řešení je však značně zjednodušující, protože ignoruje skutečnost, že texel může pokrývat větší a tvarově složitější část povrchu, přičemž jeho vlastnosti – například orientace normály nebo dopadající osvětlení – se mohou v rámci texelu výrazně lišit. Tento problém částečně řeší zvolení vyššího rozlišení lightmapy, resp. vyšší hustoty texelů. Navíc také není zajištěno, že střed texelu skutečně leží uvnitř oblasti texelu, která se překrývá s povrchem geometrie. V takovém případě by se pro danou oblast nevygeneroval žádný vzorek a v daném místě by tedy nebylo vypočteno žádné osvětlení, přestože by část povrchu do texelu zasahovala.

Z těchto důvodů je vhodné použít pokročilejší strategii vzorkování, která lépe zohledňuje skutečný tvar oblasti, v níž se texel překrývá s povrchem. Cílem je zajistit reprezentativní pokrytí texelu vzorky tak, aby bylo možné přesněji zachytit případnou variabilitu osvětlení v rámci celé jeho plochy.

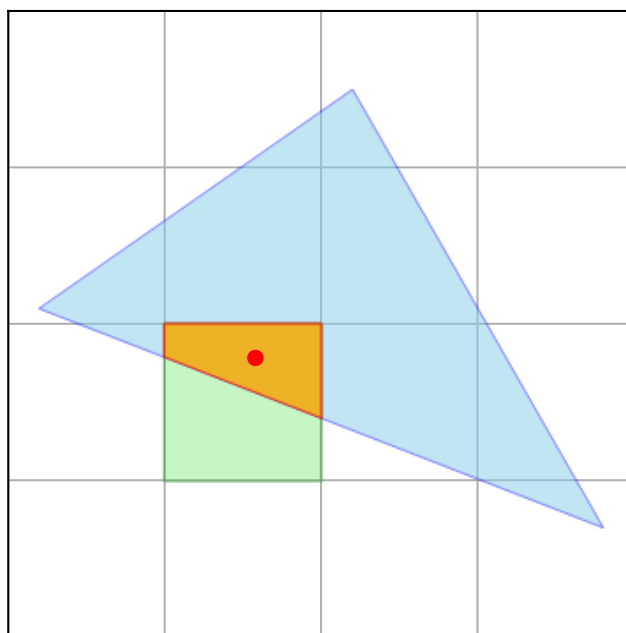
Při návrhu strategie vzorkování plochy texelu jsem vycházel z metody představené společností Electronic Arts v roce 2018 jako součást řešení pro generování lightmap v jejich proprietárním herním enginu *Frostbite* (O'Donnell 2018, s. 82). Tato metoda využívá *supersampling* jednotlivých texelů pomocí posloupnosti s nízkou diskrepancí, konkrétně Hammersleyho posloupnosti o velikosti 64 vzorků. To znamená, že pro každý texel je vytvořeno 64 vzorků (bodů), přičemž pro každý z nich se ověří, zda leží uvnitř některého z trojúhelníků v prostoru sekundárního UV mapování. Pokud ano, je bod zařazen do seznamu platných vzorků pro daný texel. Při následném vytváření primárního paprsku se z tohoto seznamu vybere jeden náhodný vzorek, přičemž jeho volba se řídí rovnoměrným rozdělením pravděpodobnosti – každý vzorek má stejnou pravděpodobnost, že bude zvolen.



Obr. 18: Grafické znázornění vytváření vzorků na povrchu geometrie pomocí posloupnosti s nízkou diskrepancí – zobrazení v UV prostoru a 3D prostoru

Zdroj: Apers a kol. (2019, s. 4)

V publikaci z roku 2019 o lightmapách ve Frostbite (Apers a kol. 2019, s. 4) je dále uvedeno i řešení situace, která nastává tehdy, kdy trojúhelník sice určitou částí zasahuje do daného texelu, avšak žádný ze vzorků vytvořených pomocí Hammersleyho posloupnosti se nenachází uvnitř trojúhelníku. K tomu může docházet například, pokud trojúhelník do texelu zasahuje pouze malou částí, nebo je-li trojúhelník natolik malý, že mívá veškeré vzorky. V takovém případě je čtvercový (nebo obdélníkový) polygon texelu omezen („ořezán“) trojúhelníkovým polygonem a nový vzorek je vytvořen v geometrickém středu jejich společného průniku.



Obr. 19: Princip omezení čtvercového polygonu texelu (zeleně) trojúhelníkovým polygonem (modře) a výsledný průnik (oranžově) s jeho geometrickým středem (červeně)

Zdroj: vlastní zpracování

Ve své první implementaci jsem metodu vzorkování plochy texelu pomocí Hammersleyho posloupnosti implementoval naivním způsobem. Pro každý trojúhelník byl nejprve spočítán jeho 2D AABB v prostoru sekundárního UV mapování. Následně byly procházeny všechny texely, u nichž se testovalo, zda se jejich plocha překrývá s AABB některého z trojúhelníků. Pokud byl takový překryv nalezen, započala fáze vzorkování, při níž se vytvářely vzorky z Hammersleyho posloupnosti a testovalo se, které z těchto vzorků leží uvnitř daného trojúhelníku.

Tento postup je z geometrického hlediska robustní a zcela přesný, avšak neefektivní – každý texel musí testovat potenciální překryv s každým z trojúhelníků, což vede k vysoké výpočetní náročnosti, která výrazně narůstá se zvyšujícím se rozlišením lightmapy a roste i se složitostí geometrie.

Možným řešením by bylo využití 2D prostorové akcelerační struktury, která by umožnila efektivněji lokalizovat relevantní trojúhelníky pro každý texel. Namísto toho jsem se však rozhodl pro odlišný přístup, jenž využívá rasterizace prostoru lightmapy do textury. To znamená, že dojde k označení („obarvení“) těch texelů, které jsou překryty některým trojúhelníkem.

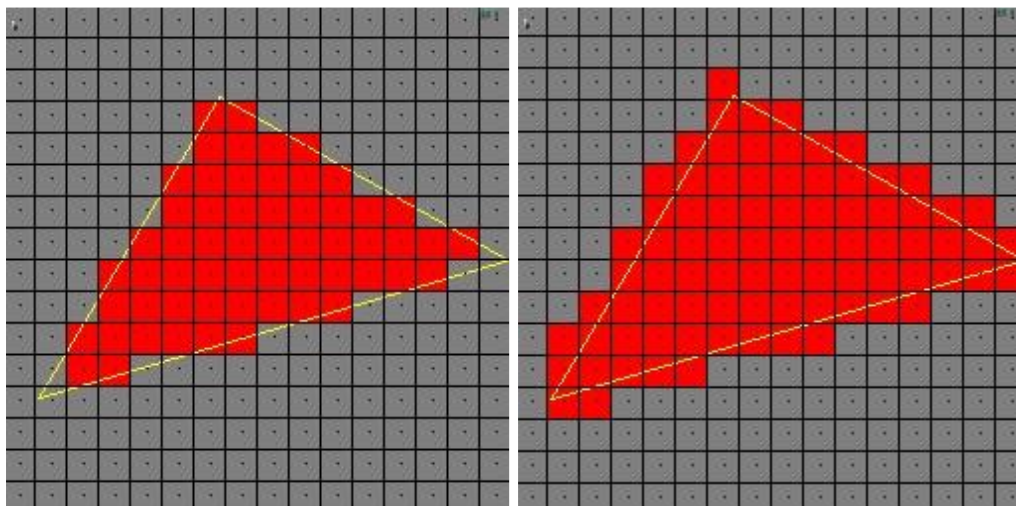
Zatímco předchozí řešení bylo z hlediska testování překryvu orientováno na texely – pro každý texel se hledaly, které trojúhelníky do něj zasahují, následující přístup je orientován na trojúhelníky – pro každý trojúhelník se zjišťuje, do kterých texelů zasahuje. Ačkoliv i v tomto případě roste výpočetní náročnost s rozlišením lightmapy, není tento růst tak výrazný jako u předchozí metody.

Další výhodou rasterizačního přístupu je možnost využití GPU, pro kterou je rasterizace jednou ze základních a vysoce optimalizovaných operací. Namísto implementace vlastního softwarového rasterizéru lze tedy použít běžné grafické API a přenechat tento úkol GPU, což navíc přináší i výrazně vyšší výpočetní výkon. Zatímco moje původní naivní, byť geometricky přesná, implementace mohla trvat i několik sekund, GPU je schopné vytvářet výstup v reálném čase – i několikrát za sekundu.

Tato metoda, jak bylo doposud popsána, však neposkytuje odpověď na otázku, které trojúhelníky do daného texelu zasahují. Toho lze však snadno docílit použitím vlastní programovatelné logiky – *fragment shaderu*, jenž ovlivňuje, jaká barva či obecně jaká data jsou do daného texelu zapsána. Dále tak výstup rasterizace nepovažujeme za barvu, ale obecně jako data. Jako jeden z výstupů si tak ukládám index trojúhelníku, který byl do daného texelu rasterizován.

Avšak i tento přístup přináší jistá úskalí. Tím prvním je, že nyní jsme omezeni na to, že do daného texelu může vždy zasahovat pouze jediný trojúhelník. To snižuje schopnost zachytit variabilitu osvětlení, pokud do texelu zasahuje více trojúhelníků. Tento jev bude nejvýraznější u lightmap s nízkým rozlišením a bude nepatrný v případě optimálního vyššího rozlišení. Toto omezení lze tak považovat za přijatelný kompromis.

Druhé, poněkud vážnější úskalí, které vyžaduje zvláštní pozornost, spočívá ve vlastnostech standardní rasterizace na GPU. Ta není *konzervativní*, což znamená, že trojúhelník se rasterizuje do texelu pouze tehdy, pokud střed texelu leží uvnitř trojúhelníku. V našem případě je však třeba postihnout všechny texely, do nichž trojúhelník jakkoli zasahuje – i ty, jejichž střed se s ním přímo nekryje. V opačném případě by některé texely, zejména na okrajích geometrie, zůstaly zcela neosvětlené. V horším případě by docházelo k tomu, že úzké části geometrie nebudou osvětleny vůbec, protože budou mít středy texelů.



Obr. 20: Porovnání výstupu standardní rasterizace (vlevo) a konzervativní rasterizace (vpravo)

Zdroj: NVIDIA (nedatováno a)

Někteří výrobci ve svých GPU nabízejí nativní podporu konzervativní rasterizace (NVIDIA nedatováno). Pokud však chceme podporovat různá GPU, potřebujeme nalézt univerzální řešení.

Na tento problém upozorňuje i Rahtenko (2018) ve svém článku, kde popisuje technické řešení vlastního neoficiálního nástroje pro generování lightmap pro herní engine *Unity*. Pro vzorkování plochy texelů využívá rovněž rasterizační přístup, přičemž naráží na stejné potíže způsobené nekonzervativní povahou standardní rasterizace.

Jím navržené řešení spočívá v emulaci konzervativní rasterizace pomocí tzv. *dilatace* – tedy rozšíření rasterizovaného výstupu o jeden texel ve všech směrech (horizontálně, vertikálně i diagonálně). Toho dosahuje opakovanou rasterizací geometrie, přičemž v každém průchodu jsou souřadnice UV mapování posunuty o jednu až dvě poloviny velikosti texelu v daném směru.

Díky tomu, že při tvorbě atlasu byla okolo každého chartu ponechána dostatečná mezera (viz 3), je při tomto způsobu rasterizace vyloučena situace, kdy by vlivem posunu došlo k nežádoucí rasterizaci do sousedního chartu. Tato mezera je záměrně o jeden texel širší, než je požadavek daný bikubickým filtrováním, které samo o sobě vyžaduje mezera dvou texelů – dohromady tedy tvoří bezpečnou mezera o velikosti tří texelů.

```
constexpr glm::vec2 kUvOffsets[] = {
    {-2, -2}, {2, -2}, {2, -2}, {-2, 2}, {2, 2}, {-1, -2}, {1, -2},
    {-2, -1}, {2, -1}, {-2, 1}, {2, 1}, {-1, 2}, {1, 2}, {-2, 0},
    {2, 0}, {0, -2}, {0, 2}, {-1, -1}, {1, -1}, {-1, 0}, {1, 0},
    {-1, 1}, {1, 1}, {0, -1}, {0, 1}, {0, 0};
```

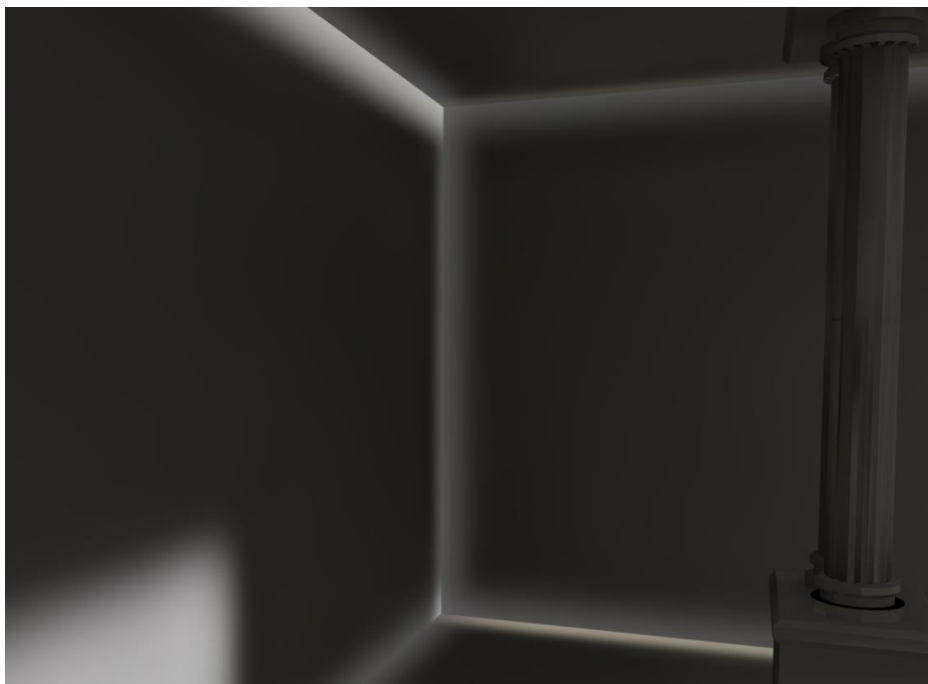
Obr. 21: Předdefinované posuny v UV prostoru, jež jsou násobeny polovinou velikosti texelu a aplikovány v několika rasterizačních průchodech za účelem emulace konzervativní rasterizace

Zdroj: Rahtenko (2018), přepsáno a upraveno pro potřeby implementace

Zásadní je pak pořadí použitých posunů. S každým dalším průchodem se část předchozího výstupu překresluje, přičemž první průchody používají posun o velikost jednoho texelu, zatímco v dalších se velikost posunu snižuje až na polovinu texelu. Poslední průchod probíhá bez jakéhokoliv posunu a odpovídá původní rasterizaci bez dilatace.

Jak již bylo dříve uvedeno, pro každý vzorek na povrchu je v mém nástroji pro účely path tracingu nezbytné znát jeho pozici, normálu a tangentu. Rasterizace pomocí GPU je tak příznivě nakloněna i této potřebě. S její pomocí jsme pro každý texel schopni mimo indexu trojúhelníku do výstupu zapisovat i pozici, normálu a tangentu, které odpovídají interpolaci vrcholových atributů daného trojúhelníku. Tato interpolace je prováděna hardwarově pomocí GPU, čímž odpadá potřeba ručního výpočtu na straně CPU.

Rahteenko (2018) jako alternativní řešení pro emulaci konzervativní rasterizace uvádí rasterizovat geometrii bez posunu a poté ji překreslit rasterizací obvodu trojúhelníku pomocí čar, tedy bez výplně. Byť toto řešení schopně napodobuje konzervativní rasterizaci, při jeho implementaci jsem narazil na zásadní problém. Středy texelů postižených při rasterizaci hran pomocí čar odpovídají bodům, které leží přesně na hraně trojúhelníku. To je zvláště problematické v místech, kde se stýká více hran – typicky v rozích. Jelikož trojúhelníková síť reprezentuje pouze povrch bez objemu, nelze jednoznačně určit, na které straně povrchu se daný vzorek nachází. V důsledku toho může docházet k úniku světla, kdy se vzorek může při výpočtu dopadajícího osvětlení ocitnout na opačné straně povrchu a uvažovat světlo z nesprávné a jinak osvětlené oblasti.



Obr. 22: Artefakty vzniklé v důsledku vzorkování na hranách trojúhelníků – v rozích místnosti dochází k nesprávnému osvětlení, které vzniká v důsledku neurčitosti, které straně povrchu daný vzorek přísluší. Tento problém se typicky vyskytuje na rozhraní dvou oblastí s rozdílným osvětlením, kde by povrch měl tvořit světelnou bariéru. Vzorky umístěné přesně na hraně však mohou vést k úniku světla mezi těmito oblastmi, což se projevuje jako nepřírozeně ostré přechody na hranách geometrie.

Zdroj: vlastní zpracování

Primárním účelem mé volby pro rasterizační přístup však bylo zjistit, zda se v daném texelu nachází některý trojúhelník, potažmo jaký. Skutečné řešení pro vzorkování texelů v mém nástroji tak probíhá následujícím způsobem. Nejprve dojde k rasterizaci lightmapového prostoru, přičemž výstup zahrnuje veškeré výše popsané atributy. Následně se prochází každý texel a zjišťuje se, zda se do něj byl rasterizován nějaký trojúhelník. Toho je docíleno snadnou

kontrolou příslušné hodnoty na výstupu proti nule, která slouží jako hodnota sentinelu. Je-li hodnota nenulová potom jde o index trojúhelníku, přičemž počátečním indexem je hodnota 1.

Následně je započata fáze vzorkování oblasti texelu pomocí 64 vzorků Hammersleyho posloupnosti a veškeré vzorky, které se nacházejí uvnitř trojúhelníku, jsou přidány do seznamu platných vzorků. Pro ověření, zda se daný vzorek nachází uvnitř trojúhelníku, je využívána předem připravená akcelerační struktura prostoru lightmapy, která umožňuje rychlý výpočet barycentrických souřadnic a tím i snadné ověření příslušnosti bodu k danému trojúhelníku.

Samotné testování všech 64 vzorků může být výpočetně náročné, a proto je nejprve provedeno optimalizační ověření: zkontroluje se, kolik rohů texelu leží uvnitř trojúhelníku. Pokud se uvnitř nacházejí všechny čtyři rohy, lze předpokládat, že celý texel je pokryt, a tím pádem jsou všechny vzorky považovány za platné. Naopak pokud se uvnitř nenachází žádný roh – což může být důsledek předchozí dilatace během rasterizace – pak není potřeba testovat žádný vzorek, protože žádný z nich platný nebude. Hammersleyho posloupnost se tedy testuje pouze v případech, kdy je některý roh uvnitř a jiný vně trojúhelníku.

V případě, že pro daný texel není nalezen žádný vzorek Hammersleyho posloupnosti, dojde k omezení jeho oblasti proti polygonu trojúhelníku, přičemž vzorek je vytvořen v jeho geometrickém středu. Vlivem dilatace však může selhat i tento proces, protože texel může být zcela vně, a tudíž s trojúhelníkem nesdílí žádnou plochu. Poté dochází na použití vzorků z rasterizovaného výstupu, což odpovídá středům texelů s případnou dilatací. K použití těchto vzorků dochází zejména v těch texelech, které se staly platnými díky dilataci.

Prozatím nebylo zmíněno, jakým způsobem jsou vytvořené vzorky reprezentovány, resp. uchovávány v paměti. Ta se liší s ohledem na typ daného vzorku. V nástroji se rozlišují dva základní typy vzorků, které se liší jak svým původem, tak způsobem reprezentace.

Potřebnými informacemi jsou souřadnice texelu v lightmapě, aby se vědělo, jakému místu na lightmapě daný vzorek odpovídá a zároveň povrchové atributy odpovídající místu na povrchu v 3D prostoru.

V případě vzorků z rasterizovaného výstupu jsou tyto atributy k dispozici již při samotné rasterizaci a přímo se ukládají jako součást vzorku. Stejným způsobem jsou reprezentovány i vzorky vytvořené v geometrických středech průniku texelu s trojúhelníkem, ovšem s tím rozdílem, že atributy jsou interpolovány na CPU na základě barycentrických souřadnic tohoto středu.

Naopak vzorky vzniklé pomocí Hammersleyho posloupnosti by v plné reprezentaci vyžadovaly uchovávat až 64 instancí vzorku pro každý texel včetně jejich atributů, což by znamenalo značnou paměťovou zátěž. Proto jsou tyto vzorky uloženy v kompaktnější podobě – jako 64bitová maska, kde každý bit označuje platnost příslušného vzorku (1 = platný, 0 = neplatný). Spolu s touto maskou je uložen také index trojúhelníku, kterému vzorky přísluší. Při samotném path tracingu pak lze náhodně zvolit některý z platných vzorků a se znalostí indexu trojúhelníku lze povrchové atributy odpovídající danému vzorku dopočítat až tehdy, kdy je to třeba.

Za účelem zvýšení kvality kvazi-náhodného vzorkování, používaného v mé implementaci path tracingu, je dále s každým texelem bez ohledu na jeho typ vzorků ukládána dvojice čísel z intervalu $<0,0; 1,0>$, náhodně generovaných s rovnoměrnou hustotou pravděpodobnosti. Ta následně slouží k dekorelaci kvazi-náhodných posloupností prostřednictvím Cranley–

Pattersonovy rotace (Cranley a Patterson 1976), tedy k potlačení případů, kdy by více různých texelů v každém kroku využívalo stejná náhodná čísla, k čemuž by v opačném případě docházelo vždy (viz 4.3.1).

Ačkoliv byl tento způsob vzorkování texelů v nástroji ponechán, neboť byl nejpřijatelnější z vyzkoušených metod, k úniku světla dochází i nadále, byť v podstatně méně patrné míře. Nelze vyloučit, že se na vzniku problému podílejí i jiné faktory než samotné vzorkování texelů. V úvahu přichází i to, jakým způsobem je prováděno vržení a sledování paprsků při výpočtu osvětlení. Při vyšších počtech vzorkování osvětlení se tento jev zmírňuje a není tolik výrazný.



Obr. 23: Projev artefaktů způsobených únikem světla do interiéru uzavřené místnosti.

Zobrazený artefakt vzniká při použití metody vzorkování texelů, která byla ponechána ve finální implementaci nástroje. Nežádoucí únik světla je patrný zejména podél svislé hrany, kde by v ideálním případě měl být prostor zcela zastíněn. Vlevo původní snímek, vpravo se zvýšeným kontrastem pro lepší viditelnost artefaktu.

Zdroj: vlastní zpracování

4.2.2 Dilatační mapa

Při vytváření vzorků uvnitř texelů lightmapy dochází ke zjištění, v kterých texelech bude třeba počítat osvětlení. Dále je však vhodné postihnout i ty texely, jež se nacházejí na vnějších okrajích chartů, tedy těch, které představují mezeru mezi charty, která byla při vytváření atlasu nastavena na hodnotu tři texely. To znamená, že v okolí každého chartu jsou vždy ponechány tři volné texely. Jeden texel je ponechán za účelem použitého rasterizačního přístupu, který využívá dilatace pro emulaci konzervativní rasterizace. Zbylé dva texely však slouží pro eliminaci artefaktů, které by vznikly v důsledku bilineárního či bikubického filtrování, jež by vzorkovalo osvětlení ze sousedních chartů a docházelo by tak k úniku světla.

Při současné situaci s vypočteným osvětlením v texelech však stále bude docházet k úniku světla, lépe řečeno stínu, pokud budeme uvažovat, že v mezerách mezi charty je hodnota osvětlení nulová. Proto je třeba tyto texely zaplnit určitou výplní, která zajistí, že se v mezerách mezi charty nachází platné hodnoty osvětlení.

Této výplně lze docílit obdobným způsobem jako při rasterizaci s dilatací – tedy rozšířením existujících platných texelů směrem ven. Jinými slovy, osvětlení z platných texelů na vnitřním okraji chartu je zkopírováno do sousedních texelů na jeho vnějším okraji.

Za tímto účelem nástroj provádí jednorázovou proceduru, která pro každý okrajový texel (v okruhu dvou texelů od hrany chartu) zaznamená cílové umístění pro výplň a zároveň určí, který texel s platnou hodnotou bude jejím zdrojem. Tato procedura je provedena ještě před samotným výpočtem osvětlení a využívá výpočetní výkon GPU prostřednictvím vlastního *compute shaderu*. Shader prochází všechny texely textury, jež je výsledkem rasterizace při vzorkování plochy texelů, a zjišťuje, zda daný texel bude obsahovat platnou hodnotu osvětlení – tedy zda do něj byl rasterizován nějaký trojúhelník. Pokud ne, může jít o texel v mezeře mezi charty. V takovém případě prohledá okolí v okruhu dvou texelů, přičemž hledá nejbližší sousední texel s platnou hodnotou. Jakmile jej najde, je jisté, že se jedná o mezeru a pro aktuální texel do výstupní textury uloží jeho relativní posun vůči platnému vzorku.

Tento výstup je následně zpracován na straně CPU, kde dochází ke vzniku tzv. *dilatační mapy*. Ta již obsahuje pouze dvojice – absolutní indexy cílových texelů, které je třeba vyplnit, a odpovídajících zdrojových texelů, z nichž se má hodnota kopírovat. Výsledná dilatační mapa pak slouží jako podklad pro samotnou dilataci osvětlení v rámci postprocessingu (viz 4.4).

4.3 Výpočet osvětlení

Po konstrukci akceleračních struktur, shromáždění vzorků pro primární paprsky a se znalostí jejich mapování do jednotlivých texelů lightmapy je dále možné přistoupit k samotnému výpočtu osvětlení.

Nástroj umožňuje výpočet globálního osvětlení, jež zahrnuje jak přímé, tak nepřímé osvětlení. Pro výpočet nepřímého osvětlení je využíváno jednosměrného zpětného path tracingu, který je prováděn na CPU.

Kvalita vypočteného osvětlení je do jisté míry ovlivněna rozlišením atlasu, které uživatel zvolil při jeho tvorbě a zároveň parametry, které uživatel volí při samotném výpočtu lightmap. Mezi tyto parametry patří:

- **Typ lightmapy**

Určuje, jakým způsobem bude výsledné osvětlení reprezentováno a ovlivňuje možnosti rekonstrukce osvětlení za běhu – nesměrové lightmapy nelze využít společně s technikou normal mappingu.

Nesměrová – ukládá intenzitu záření dopadající na povrch podél normály

Formáty:

- Hodnota intenzity záření

Směrová – ukládá směrovou distribuci intenzity záření v hemisféře nad povrchem

Formáty:

- Projekce intenzity záření na SH třetího řádu

- **Počet vzorků**

Kolik nezávislých vzorkování osvětlení proběhne pro každý jednotlivý texel – výsledná hodnota texelu je určena jejich váženou akumulací.

- **Maximální počet odrazů**

Omezuje maximální počet odrazů od povrchů, které mohou nastat při sledování cesty paprsku, než dojde k jeho umělému ukončení.

Kolikrát se nanejvýš může paprsek světla odrazit od povrchů, než dojde k umělému ukončení jeho cesty. Je-li hodnota nulová, dochází pouze k výpočtu přímého osvětlení.

- **Počet vláken**

Určuje, kolik paralelních vláken bude použito při výpočtu osvětlení. Tento parametr má vliv pouze na rychlost výpočtu a na kvalitu výsledku nemá dopad.

- **Odchylka paprsků**

Určuje vzdálenost, o kterou je počáteční bod paprsku posunut ve směru normály, aby se zabránilo tzv. *self-intersection* artefaktům, kdy by povrch mohl „stínit sám sobě“.

Je-li nulová, využívá se při výpočtu jednak výchozích odchylky (0,0001) a odchylek určených automaticky pomocí metody navržené Wächterem a Binderem (2019).

- **Počáteční odraz ruské rulety**

Určuje počet odrazů paprsku, které jsou vždy vykonány před tím, než se začne uplatňovat technika *ruské rulety*.

- **Maximální počet průchodů paprsků skrze průsvitné povrchy**

Udává, kolikrát může paprsek při výpočtu přímého osvětlení projít průsvitnými materiály. Při dosažení limitu je světelný zdroj z pohledu osvětlovaného bodu považován za zakrytý.

Se znalostí těchto parametrů lze popsat, jakým způsobem je v nástroji implementován progresivní výpočet osvětlení.

Na rozdíl od vytváření vzorků pro primární paprsky, které částečně využívá výpočetního výkonu GPU, výpočet osvětlení probíhá čistě na CPU. Samotný systém je pak navržen tak, aby umožňoval paralelní výpočet na více vláknech. Nejdříve tak dochází k rozdělení celé práce do více paralelních úloh, přičemž počet úloh odpovídá počtu vláken, který uživatel nastavil v konfiguraci.

Jelikož výpočet osvětlení v každém texelu probíhá nezávisle na ostatních, lze celý proces snadno paralelizovat. Texely jsou rovnoměrně rozděleny mezi dostupná vlákna, kdy každé z nich zpracovává jinou podmnožinu texelů. Zpracování jednotlivých podmnožin pak probíhá souběžně na všech vláknech.

Dále je každému vláknu třeba zpřístupnit samotné vzorky, interní reprezentaci scény s akceleračními strukturami a zároveň paměť, kam bude moci ukládat své průběžné výsledky. Pro tyto účely je vytvořen kontext, který představuje objekt, jenž ukládá data sdílená napříč vlákny. Samotné průběžné výsledky se pak v paměti udržují ve dvou podobách.

Společným pro obě podoby je způsob, jakým se hodnoty vypočteného osvětlení uchovávají v paměti. Každý texel je reprezentován jako čtyřkanálový vektor (RGBA)¹⁴, přičemž každý kanál je uložen jako 32bitové číslo s plovoucí desetinnou čárkou. V této reprezentaci jsou pak hodnoty osvětlení ukládány ve *vysokém dynamickém rozsahu* (HDR), jenž umožňuje detailně zachytit široké spektrum intenzit osvětlení, a tím zachovat věrný kontrastní poměr mezi velmi tmavými a velmi jasnými oblastmi ve scéně. Nástroj však podporuje různé formáty lightmap, což

¹⁴ Alpha kanál ve své podstatě potřebný není, neboť světlo nemá průhlednou složku. V nástroji je však používán, protože čtyřkanálový formát (RGBA) je běžně podporován při přenosu dat do paměti GPU.

znamená, že texel může jako osvětlení ukládat např. jedinou hodnotu (koeficient) intenzity záření na kanál pro každý texel v případě nesměrových lightmap nebo devět koeficientů na kanál pro distribuci intenzity záření v případě směrových lightmap využívajících SH třetího řádu. Z tohoto důvodu jsou surová data organizována do vrstev, kde každá vrstva odpovídá jednomu koeficientu daného formátu.

První podobou průběžných výsledků jsou *surová texturová data* lightmapy reprezentovaná jako lineární pole RGBA vektorů o velikosti odpovídající celkovému počtu texelů. Tato forma je pak zpřístupněna vyšším úrovním nástroje pouze pro čtení, které ji mohou využít pro získání nejaktuálnější podoby průběžně vznikající lightmapy. Do daného texelu pak může vždy zapisovat pouze jediné vlákno, kterému byl příslušný texel přidělen. Z tohoto důvodu není třeba čtení či zápis surových dat žádným způsobem synchronizovat ani se zabývat možným blokováním při přístupu k datům. Ve skutečnosti sice může dojít ke stavu, kdy je hodnota texelu zapsána pouze částečně, přičemž by byla zároveň čtena jiným vláknem. V praxi je však tento problém pro účely zobrazení průběžně vznikající lightmapy zcela zanedbatelný. Surová data si neuchovávají žádný stav a při získání nové hodnoty pro daný texel jsou průběžně přepisována.

Druhou je tzv. *akumulační podoba*, která reaguje na potřebu uchovávat a postupně akumulovat výsledky jednotlivých vzorků světla, jež přispívají k výsledné hodnotě pro daný texel lightmapy. Pro tyto účely slouží tzv. *akumulátor vzorků texelu* (dále jen akumulátor), jenž poskytuje rozhraní pro přírůstkovou akumulaci přímého a nepřímého osvětlení při vzorkování osvětlení, přičemž se stará o správný průběh Monte Carlo integrování. Akumulátor si tak udržuje stav ve formě akumulovaného přímého a nepřímého osvětlení. Obě složky osvětlení jakož i počty akumulovaných vzorků jsou udržovány zvlášť, neboť se tyto počty mohou lišit. To vychází z optimalizace, kterou lze provádět při výpočtu přímého osvětlení pro primární paprsky s pevnou pozicí – tedy těch, které se nevolí náhodně ze vzorků Hammersleyho posloupností. V případě vzorků s pevnou pozicí by docházelo k opakovanému výpočtu přímého osvětlení, které by vedlo na stejný výsledek, proto jej stačí počítat pouze jednou.

Jako v případě surových texturových dat, se rovněž i způsob akumulace výsledků liší s ohledem na formát osvětlení v lightmapě. Za tímto účelem byly navrženy různé konkrétní specializace rozhraní akumulátorů pro každý specifický formát. Počet akumulátorů potřebných pro výpočet lightmapy odpovídá počtu texelů, v kterých je vzorkováno osvětlení. To znamená, že metody příslušného rozhraní jsou volány při každém vzorkování texelu. Přestože by rozhraní akumulátoru bylo možné implementovat pomocí klasického polymorfismu a využít dynamický výběr metod za běhu (*dynamic dispatch*), takové řešení by s sebou neslo dodatečnou výpočetní režii.

Z tohoto důvodu jsem zvolil implementaci založenou na statickém polymorfismu se statickým výběrem (*static dispatch*) využívajícím šablon (*templates*), resp. *generického programování*, kdy je konkrétní typ akumulátoru znám již v době kompilace. To nejen eliminuje režii výběru metody za běhu, ale zároveň umožňuje kompilátoru provádět optimalizace jako např. *inlining* volaných metod.

Vzhledem k tomu, že typ lightmapy, resp. formát osvětlení v lightmapě má přímý vliv jak na počet vrstev surových texturových dat, tak i na typ použitých akumulátorů, je volba konkrétní specializace provedena již při vytváření společného kontextu. Tento kontext pak předává typ lightmapy, jenž je znám při kompilaci, jako šablonový parametr surovým texturovým datům

lightmapy i akumulátorům. To umožňuje, aby většina rozhodnutí ohledně konkrétní implementace byla provedena staticky již při kompilaci. Dynamicky je tedy rozhodováno pouze jednou – při vytváření kontextu a spouštění paralelních vláken na základě uživatelského nastavení typu lightmapy. Veškerá další logika, která se od typu lightmapy odvíjí, je tak pevně určena již při kompilaci.

```
template <typename T>
concept TexelSampleAccumulator =
    requires(T t, const glm::vec3& dir, const glm::vec3& radiance,
             const glm::vec3& irradiance, const glm::vec2& random,
             ImageLayers<T::kCoefficientCount>& output, size_t offset) {
        {
            T::sample_hemisphere_tangent_space(random)
        } -> std::same_as<glm::vec3>;
        {
            t.add_ambient_light(irradiance)
        } -> std::same_as<void>;
        {
            t.add_direct_light(dir, radiance)
        } -> std::same_as<void>;
        {
            t.increment_direct_sample_count()
        } -> std::same_as<void>;
        {
            t.add_indirect_light(dir, radiance)
        } -> std::same_as<void>;
        {
            t.increment_indirect_sample_count()
        } -> std::same_as<void>;
        {
            t.store_to_texture_data(output, offset)
        } -> std::same_as<void>;
    };
```

Obr. 24: Společné rozhraní akumulátoru. Konkrétní specializace se liší podle formátu osvětlení v lightmapě.

Zdroj: vlastní zpracování

Hlavní odlišností mezi konkrétními specializacemi jsou zejména formát dat, který se využívá pro akumulaci a způsob, jakým se volí směry primárních paprsků při vzorkování hemisféry nad osvětlovaným bodem pro odhad nepřímého osvětlení. Druhá odlišnost ze zmíněných pak není dána ani tak formátem jako takovým, jako spíše, zda je daný formát určený pro nesměrové či směrové lightmapy. Tato binární klasifikace je pak rozhodující při výběru strategie pro vzorkování hemisféry a vážení vzorků při Monte Carlo integrování.

U nesměrových lightmap, kde je normála povrchu používána pro stínování známa předem, se používá vzorkování hemisféry vážené kosinem. Příspěvek do nepřímého osvětlení je v podobě záře, která do bodu přichází z okolního prostředí v určitém vzorkovaném směru. Pro uložení výsledku do nesměrové lightmapy je však třeba určit, jaká část této záře se odrazí ve směru povrchové normály, a tuto hodnotu integrovat přes hemisféru za účelem získání výsledné intenzity záření. Z definice rovnice odrazu a informací z kapitoly 1.5.2 však vyplývá, že vážení vzorků pro Monte Carlo integrování lze zjednodušit, neboť váha vzorku je přímo úměrná chování lambertovské BRDF modulované členem $(n \cdot l)^+$. Díky tomu každý vzorek přicházející záře již přímo odpovídá záři odražené od povrchu.

Naopak u směrových lightmap není normála povrchu, která se používá pro stínování, předem známa, neboť ta se při vykreslování vzorkuje z normálové mapy. Jediným známým předpokladem je skutečnost, že se normála bude nacházet v rámci hemisféry nad osvětlovaným bodem. To znamená, že aby odhad Monte Carlo integrování nebyl zaujatý, je správné každému vzorku – směru v hemisféře přikládat stejnou váhu. Ta odpovídá polovině povrchu jednotkové koule – 2π .

S ohledem na formát lightmapy se liší i konkrétní data, která se zapisují do texturových dat. Těm pak musí být uzpůsoben i shader používaný při vykreslování, který se stará o mapování lightmapy na povrch a rekonstrukce osvětlení z ní. V případě nesměrových lightmap hodnota v texelu odpovídá intenzitě záření. Výsledná barva stínovaného bodu je pak dána násobením této hodnoty s barvou albeda.

Pro účely směrových lightmap se ve formátu SH třetího řádu do texelů ukládají koeficienty SH, představující projekci záře přicházející do bodu z okolního prostředí. Tato projekce v sobě již zahrnuje konvoluční koeficienty $kA0$, $kA1$ a $kA2$ (viz příslušné konstanty v Obr. 25), které zajišťují, že při následném vzorkování hodnoty z SH v libovolném směru, bude výstup odpovídat konvoluci přicházející záře s kosinovým faktorem $(n \cdot l)^+$. Tato hodnota pak odpovídá intenzitě záření, již je zapotřebí k určení výsledné barvy osvětlovaného bodu.

Při vykreslování je pro získání intenzity ve směru povrchové normály (vzorkované z normálové mapy) zapotřebí nejprve provést projekci normály na báze funkce SH a následně provést skalární součin těchto koeficientů s koeficienty vzorkovanými z lightmapy. Jestliže však uvážíme, že projekční koeficienty $kC0$ – $kC4$ (viz příslušné konstanty v Obr. 25) jsou konstantní a předem známé, nabízí se je zahrnout do koeficientů uložených v lightmapě. Nástroj proto při projekci příchozí záře na SH aplikuje projekční koeficienty dvakrát – jednou k projekci příchozí záře a podruhé k budoucí projekci povrchové normály. Výsledné koeficienty jsou následně uloženy v lightmapě. Tento přístup sice na první pohled nemusí působit intuitivně, protože v sobě zahrnuje koeficienty, které se vztahují k projekci normály, jež nastane až při vykreslování – nicméně tímto umožňuje zjednodušit výpočty prováděné při vykreslování a rekonstrukci osvětlení z lightmapy.

Projekční i konvoluční koeficienty pro SH třetího řádu byly převzaty z práce Ramamoorthiho a Hanrahana (2001).

```
auto ShL2::project_onto_sh_l2_preconvolved_irradiance(const glm::vec3& direction) -> ShL2
{
    constexpr float kC0 = 0.282095f;
    constexpr float kC1 = 0.488603f;
    constexpr float kC2 = 1.092548f;
    constexpr float kC3 = 0.315392f;
    constexpr float kC4 = 0.546274f;

    constexpr float kA0 = std::numbers::pi_v<float>;
    constexpr float kA1 = (2.0 * std::numbers::pi_v<float>) / 3.0;
    constexpr float kA2 = std::numbers::pi_v<float> / 4.0;

    ShL2 sh{
        kC0 * kC0 * kA0, // l=0, m=0

        (kC1 * kC1 * kA1) * direction.y, // l=1, m=-1
        (kC1 * kC1 * kA1) * direction.z, // l=1, m=0
        (kC1 * kC1 * kA1) * direction.x, // l=1, m=1

        (kC2 * kC2 * kA2) * direction.x * direction.y, // l=2, m=-2
        (kC2 * kC2 * kA2) * direction.y * direction.z, // l=2, m=-1
        (kC3 * kC3 * kA2) * (3.0f * direction.z * direction.z - 1.0f), // l=2, m=0
        (kC2 * kC2 * kA2) * direction.x * direction.z, // l=2, m=1
        (kC4 * kC4 * kA2) * (direction.x * direction.x - direction.y * direction.y) // l=2, m=2
    };

    return sh;
}
```

Obr. 25: Kód sloužící pro projekci příchozí záře v daném směru do reprezentace předpočítaného osvětlení pomocí SH třetího řádu, používané nástrojem. Samotná zář není v této části kódu zohledněna, její projekce však odpovídá násobení záře s výsledkem vráceným touto metodou.

Zdroj: vlastní zpracování

```

vec3 evaluate_sh_l2_preconvolved_irradiance(vec3 direction, ShL2Vec3 sh_irradiance)
{
    ShL2 sh_local;
    sh_local.coefficients[0] = 1.0; // l=0, m=0
    sh_local.coefficients[1] = direction.y; // l=1, m=-1
    sh_local.coefficients[2] = direction.z; // l=1, m=0
    sh_local.coefficients[3] = direction.x; // l=1, m=1
    sh_local.coefficients[4] = direction.x * direction.y; // l=2, m=-2
    sh_local.coefficients[5] = direction.y * direction.z; // l=2, m=-1
    sh_local.coefficients[6] = (3.0 * direction.z * direction.z - 1.0); // l=2, m=0
    sh_local.coefficients[7] = direction.x * direction.z; // l=2, m=1
    sh_local.coefficients[8] = (direction.x * direction.x - direction.y * direction.y); // l=2,
    m=2

    vec3 result = vec3(0.0);
    for (int i = 0; i < 9; ++i)
    {
        result += sh_irradiance.coefficients[i] * sh_local.coefficients[i];
    }

    return result;
}

```

Obr. 26: Kód ve fragment shaderu, který při vykreslování slouží pro získání intenzity záření v daném směru povrchové normály. Kód provádí skalární součin koeficientů dané povrchové normály s koeficienty SH třetího řádu obsahujícími předkonvolovanou intenzitu záření zahrnujícími i koeficienty pro projekci povrchové normály na báze funkce SH. Koeficienty předkonvolované intenzity záření jsou získány vzorkováním lightmapy v příslušném místě povrchu.

Zdroj: vlastní zpracování

4.3.1 Paralelní výpočet osvětlení na více vláknech

Každé vlákno v rámci výpočtu lightmapy zpracovává přidělenou podmnožinu texelů. Pro každý z těchto texelů provádí určitý počet vzorkování osvětlení na základě uživatelem zvoleného počtu vzorků. Samotný výpočet probíhá ve dvou vnořených smyčkách – vnější iteruje přes počet vzorkování osvětlení, zatímco vnitřní prochází texely přiřazené danému vláknu.

V každé iteraci je pro daný texel připraven výpočetní kontext zahrnující pozici vzorku na povrchu cílového objektu, normálu, tangentu a posun používaný pro dekoraci kvazi-náhodných čísel. Postup pro zpracování daného texelu se dále liší s ohledem na typ jeho vzorků pro primární paprsky (vzorky Hammersleyho posloupnosti nebo vzorek s pevnou pozicí).

U vzorků s pevnou pozicí jsou tyto atributy již předem určeny a obsaženy v datové struktuře společně se vzorkem.

V případě vzorků Hammersleyho posloupnosti dochází k náhodnému výběru jednoho z platných vzorků, který se řídí rovnoměrnou hustotou pravděpodobnosti. V místě daného vzorku jsou následně interpolovány vrcholové atributy, kterými je dána skutečná pozice, normála a tangenta na příslušném místě vzorku na povrchu. Ke zjištění platných vzorků je využito bitové masky, která ukládá příznak, zda je vzorek Hammersleyho posloupnosti na příslušné pozici bitu platný či neplatný. Ke zjištění celkového počtu platných vzorků je využito funkce `std::popcount` ze standardní knihovny C++. Pro získání pozice n-tého platného vzorku v bitové masce je využito optimalizované funkce, převzaté z implementace zveřejněné uživateli Toby Speight a grek40 na fóru Stack Overflow (2017).

Následně je pro daný vzorek sestavena transformační matice ortogonální báze tečného prostoru, jež je definována trojicí vzájemně kolmých vektorů – normály, tangenty a bitangenty, přičemž bitangenta odpovídá vektorovému součinu normály a tangenty.

Dále přichází na řadu použití akumulátoru vzorků pro daný texel, jehož chování je určeno konkrétním typem lightmapy, která se právě generuje. Pomocí jeho metody *sample_hemisphere_tangent_space()* je v lokálním tečném prostoru vytvořen jeden vzorek – směr v hemisféře, který bude sloužit pro odhad nepřímého osvětlení. Jako vstup do této metody je využíváno kvazi-náhodných čísel generovaných posloupností s nízkou diskrepancí – konkrétně Haltonovou posloupností. Při jejím použití dochází k rovnoměrnějšímu pokrytí vzorkovaného prostoru než při použití pseudonáhodných čísel. To se následně při Monte Carlo integrování projevuje snížením rozptylu odhadu a vede k rychlejší konvergenci. Díky tomu lze při stejném počtu vzorkování osvětlení dosáhnout kvalitnějšího odhadu nepřímého osvětlení s nižší úrovní šumu, než kdyby bylo využito pseudonáhodných čísel.

Jako index do posloupnosti je používán aktuální index vzorkování osvětlení. To však znamená, že Haltonova posloupnost generuje stejná náhodná čísla pro každý texel při daném indexu vzorkování osvětlení. Na tento problém poukazuje i O'Donnell (2018, s. 84), přičemž uvádí, že posloupnosti náhodných čísel pro každý texel by měly být dekorelovány. To by v opačném případě vedlo k nežádoucím artefaktům – všechny primární paprsky by směřovaly ve stejném směru, a odhady nepřímého osvětlení by tak byly příliš podobné napříč celou lightmapou.

V nástroji je tento problém řešen použitím *Cranley-Pattersonovy rotace* (Cranley a Patterson 1976), kdy je každému texelu přidělena dvojice náhodných posunů, které jsou přiřčeny k hodnotě získané z Haltonovy posloupnosti. Skutečná hodnota náhodného čísla je pak dána zbytkem po dělení tohoto součtu hodnotou jedné.

Vzorkovaný směr v tečném prostoru je následně s využitím ortogonální báze transformován do světového prostoru a využit pro odhad nepřímého osvětlení metodou path tracingu (viz 4.3.4). Výsledná hodnota je následně akumulována pomocí metody *add_indirect_light()* rozhraní akumulátoru. Je-li uživatelem zvolený maximální počet odrazů nulový, odhad nepřímého osvětlení se přeskočí a místo něj se jako jeho hrubá jednobarevná aproximace použije barva oblohy – tzv. *ambientní světlo*. K tomu slouží metoda *add_ambient_light()* rozhraní akumulátoru. V obou případech se počet vzorků nepřímého osvětlení zvýší o hodnotu jedna.

Nezávisle na tom vždy dochází alespoň k výpočtu přímého osvětlení (viz 4.3.3). Pomocí vlastní akcelerační struktury *light grid* jsou vyhledány všechny relevantní světelné zdroje, které potenciálně působí na daný bod na povrchu. Pro každý z nich je vypočítán jeho příspěvek do osvětlení, který je následně akumulován metodou *add_direct_light()* rozhraní akumulátoru. Akumulace všech těchto příspěvků pak odpovídá jednomu vzorkování přímého osvětlení. To v akumulátoru odpovídá inkrementování počtu vzorků přímého osvětlení o hodnotu jedna.

Pro vzorkování přímého osvětlení platí také jedna výjimka. U texelů se vzorkem s pevnou pozicí se přímé osvětlení počítá pouze při prvním průchodu, protože při dalších iteracích by vedlo ke stejnému výsledku.

Po dokončení jedné iterace vzorkování osvětlení v daném texelu jsou průběžné výsledky akumulátoru zprůměrovány podle aktuálního počtu vzorků – zvlášť pro přímé a nepřímé osvětlení – a jejich součet je následně zapsán do surových texturových dat.

4.3.2 Sledování paprsků

Při výpočtu přímého i nepřímého osvětlení dochází ke sledování paprsků, při kterém je třeba hledat a testovat průsečíky paprsků s geometrií. K tomuto účelu je využita akcelerační struktura zprostředkovaná knihovnou *Intel Embree* (Intel nedatováno a). Při úspěšném nalezení průsečíku je pak třeba v místě zásahu vyhodnotit i odpovídající atributy povrchu, které jsou získány z bufferů (některé přímo, jiné interpolací vrcholových atributů), jež byly dříve připraveny během konstrukce akcelerační struktury pro hledání průsečíků s paprsky (viz 4.1.1).

Při výpočtu osvětlení dochází k vržení paprsků, jejichž počátek leží přímo na povrchu objektu. Vzhledem k tomu, že povrchy trojúhelníkové sítě jsou nekonečně tenké, je třeba jednoznačně určit, na které straně povrchu počátek paprsku leží. Zároveň je třeba zohlednit numerické nepřesnosti při výpočtu průsečíků. Proto je před vržením paprsku jeho skutečný počátek mírně posunut ve směru povrchové normály (viz odchylka paprsků **Chyba! Nenalezen zdroj odkazů.**), aby se předešlo tomu, že by povrch stínil sám sobě.

4.3.3 Výpočet přímého osvětlení

Příspěvek přímého osvětlení z jednoho světelného zdroje je v nástroji počítán pomocí techniky explicitního vzorkování světla (*explicit light sampling*). Vzhledem k tomu, že nástroj podporuje pouze bezrozměrné světelné zdroje, byla volba této techniky přirozená.

Pro osvětlovaný bod na povrchu se nejprve určí směr ke světelnému zdroji a z tohoto bodu je vržen tzv. *stínový paprsek* ve směru ke světlu. Není-li paprsek blokován žádnou překážkou, považuje se světelný zdroj za viditelný a jeho přínos je započítán. V opačném případě je přínos nulový.

Algoritmus podporuje také průsvitné materiály – pokud stínový paprsek zasáhne povrch s určitou mírou průsvitnosti, nová propustnost světla je dána lineární interpolací mezi předchozí propustností a albedem povrchu, přičemž váhou je průsvitnost povrchu.

Paprsek následně pokračuje dál, dokud není zcela pohlcen, nedosáhne světelného zdroje, nebo není překročen uživatelem definovaný limit maximálního počtu průniků skrze průsvitné materiály.

Výsledný příspěvek světla je pak určen intenzitou světelného zdroje, která vychází z definice daného typu světelného zdroje. Daný příspěvek odpovídá záři emitované světelným zdrojem ve směru k osvětlovanému bodu.

4.3.4 Výpočet nepřímého osvětlení

Výpočet nepřímého osvětlení je v nástroji realizován pomocí metody jednosměrného zpětného path tracingu. Implementace podporuje pouze difúzní odrazy modelované lambertovskou BRDF a zároveň umožňuje průchod paprsků skrze průsvitné materiály.

Simulace cesty paprsku začíná v osvětlovaném bodě na povrchu cílového objektu, odkud je vržen primární paprsek do scény. V průběhu jeho cesty se akumuluje přenášená zář, kterou paprsek získává při své cestě a zároveň propustnost světla, která se při každé interakci mění v závislosti na vlastnostech povrchu daných jeho BSDF.

Při každém vržení paprsku se nejprve testuje jeho průnik s geometrií ve scéně. Pokud paprsek žádný povrch nezasáhne, je k přenášené záři přičten příspěvek oblohy a sledování cesty paprsku zde končí.

V opačném případě jsou v místě zásahu získány materiálové vlastnosti povrchu a následuje výpočet přímého osvětlení pomocí explicitního vzorkování světelných zdrojů (viz 4.3.3). S pomocí *light grid* je zjištěno, které světelné zdroje potenciálně na místo zásahu působí, a pro každé z nich je následně vypočítána jeho interakce s povrchem. Interakce se řídí podle lambertovské BRDF s podporou průsvitnosti. Výsledek interakce je následně vynásoben aktuální hodnotou propustnosti a akumulován do paprskem přenášené záře.

Nová hodnota propustnosti světla se získá lineární interpolací mezi předchozí propustností a albedem povrchu, přičemž váhou je průsvitnost povrchu.

Následuje rozhodnutí o dalším směru paprsku. Pokud je povrch částečně průsvitný, je s pravděpodobností odpovídající průsvitnosti materiálu rozhodnuto o tom, zda paprsek pokračuje průchodem skrze povrch ve stejném směru, nebo se odrazí.

V případě odrazu je nový směr náhodně vybrán pomocí kosinem váženého vzorkování hemisféry nad zasaženým bodem. Pro vzorkování se využívá pseudonáhodných čísel s rovnoměrnou hustotou pravděpodobnosti.

Aby se předešlo nekonečnému sledování paprsků, uživatel definuje maximální počet odrazů. Zároveň je možné nastavit i počet odrazů, po jehož dosažení se aktivuje technika *ruské rulety*, která s ohledem na aktuální propustnost snižuje pravděpodobnost dalšího sledování cesty. Je-li očekávaný příspěvek k výslednému osvětlení zanedbatelný, cesta paprsku je předčasně ukončena.

Výsledná přenášená zář pak odpovídá nepřímému osvětlení, které do osvětlovaného bodu na povrchu cílového objektu přichází v opačném směru primárního paprsku.

4.4 Postprocessing

K fázi postprocessingu dochází buď volitelně před zobrazením průběžného výsledku během výpočtu generované lightmapy nebo vždy po jeho úplném dokončení. Samotný postprocessing pak sestává ze dvou dílčích kroků – dilatace a denoisingu.

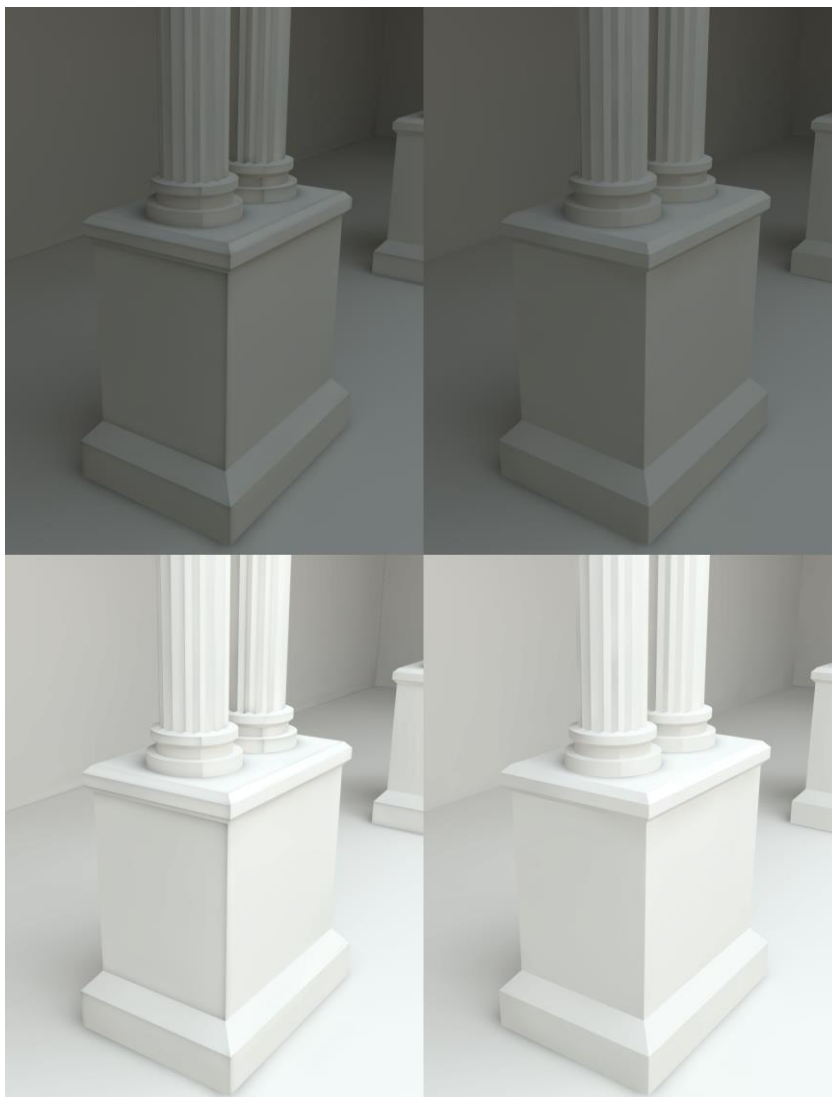
Při dilataci dochází k rozšíření vypočteného osvětlení do mezer mezi charty, které byly ponechány za účelem eliminace artefaktů při filtrování textur při vzorkování osvětlení z lightmapy. Při dilataci je využíváno dilatační mapy, která byla vytvořena v rámci preprocessingu (viz 4.2.2). Ta nese informaci o dvojicích uchovávajících absolutní index cílového texelu a zdrojového texelu, určujících, kam a odkud se má osvětlení kopírovat, resp. rozšiřovat.

Druhým krokem, který následuje po dilataci je denoising, jehož účelem je potlačení šumu v lightmapě, který je přítomen v důsledku povahy Monte Carlo integrování při výpočtu osvětlení metodou path tracingu. Nástroj využívá denoiseru implementovaného v knihovně *Open Image Denoise* od společnosti Intel (nedatováno c), který pro denoising využívá vah natrénovaných pomocí strojového učení. Knihovna poskytuje více typů filtrů, přičemž v nástroji je použit filtr „RTLightmap“, jenž je specializován na potlačení šumu v lightmapách.

Vyžaduje-li konkrétní formát lightmapy více vrstev, resp. více koeficientů na texel, provádí se obě operace pro každou vrstvu koeficientů zvlášť.

Následující obrázky představují porovnání průběžných výstupů při aktivaci/deaktivaci jednotlivých kroků postprocessingu.

Pro vzorkování lightmapy je ve všech případech využito bikubického filtru. V případě deaktivované dilatace si povšimněte chybějícího osvětlení na hranách povrchu geometrie, resp. švů chartů. V případě deaktivovaného denoisingu si povšimněte přítomnosti šumu.



Obr. 27: Lightmapa na povrchu objektu před dilatací (vlevo) a po dilataci (vpravo). Povšimněte si chybějícího osvětlení podél hran v místě švů (tmavé hrany). U spodních snímků byl pro lepší viditelnost upraven kontrast.

Zdroj: vlastní zpracování



Obr. 28: Průběžně generovaná lightmap bez využití denoisingu.

Zdroj: vlastní zpracování



Obr. 29: Průběžně generovaná lightmapa s využitím denoisingu.

Zdroj: vlastní zpracování

5 Souborový formát scény

Aby byl nástroj snadno použitelný, je vhodné podporovat některý ze standardních formátů pro ukládání 3D modelů. Tím je zároveň zajištěna kompatibilita s běžně používanými modelovacími programy, které do těchto formátů exportují, a zároveň odpadá potřeba navrhovat vlastní formát s nutností ručního převodu či úprav dat před jejich použitím v nástroji.

Z tohoto důvodu jsem se rozhodl zvolit standardizovaný souborový formát glTF (The Khronos Group nedatováno a). Pro načítání dat z tohoto formátu pak nástroj používá knihovnu *fastgltf* (Apeler nedatováno). Přestože tento formát umožňuje i ukládání světelných zdrojů, jejich parametrizace se neztotožňuje s modelem světelných zdrojů, které používá nástroj. Nástroj proto tento formát používá pouze pro načítání geometrických objektů, materiálů a případných textur.

Vzhledem k vlastní parametrizaci světelných zdrojů a dalších informací, které je ukládat společně se scénou, nástroj používá ještě další vlastní soubor typu JSON. Tento soubor pak ukládá data, která jsou specifická pro scénu, tak jak je reprezentována v nástroji. Tato data zahrnují definice světelných zdrojů či příslušnost vygenerovaných lightmapových atlasů a samotných lightmap k určitému geometrickému objektu. O serializaci a deserializaci tohoto souboru se pak stará knihovna *JSON for Modern C++* (Lohmann nedatováno).

Kompletní popis všech náležitostí souborového formátu scény však není vzhledem k cíli práce podstatný. Proto se dále zaměřuji především na lightmapový atlas a lightmapy.

Každý lightmapový atlas a lightmapa jsou v nástroji jednoznačně identifikovány pomocí *univerzálního unikátního identifikátoru* (UUID), jež se vytvářejí prostřednictvím knihovny *stduuid* (Bancila nedatováno). Samotný lightmapový atlas a lightmapa však představují pro scénu externí zdroj dat. JSON scény se tak na tato data pouze odkazuje pomocí jejich UUID, přičemž samotná data lightmapového atlasu i lightmap jsou ukládána zvlášť uvnitř adresáře „lightmap_data“ v kořenovém adresáři scény. V případě lightmap se pak v JSONu scény ukládá také jakého formátu daná lightmapa je, z čehož vyplývá, kolik koeficientů daný formát využívá.

Pro uložení lightmapového atlasu nástroj využívá jednoduchého vlastního formátu, který binárně ukládá kompaktní reprezentaci lightmapového atlasu tak, jak byla popsána v kapitole 3.

Souborový formát lightmap si však zaslouhuje vyšší pozornost, proto je mu věnována samostatná kapitola.

5.1 Souborový formát lightmap

Přestože je lightmapa ve své podstatě texturou, kterou lze uložit v libovolném rastrovém grafickém formátu, každý z těchto formátů má svá specifika, výhody i omezení – zejména z hlediska komprese a barevné věrnosti. Problematikou komprese jsem se však příliš nezabýval a pro ukládání lightmap jsem zvolil formát PNG, který reprezentuje barvy pomocí tří (RGB) nebo čtyř (RGBA) kanálů. V případě lightmap si vystačíme pouze s třemi, jelikož světlo nevyžaduje průhlednost.

Problém však spočívá v omezeném rozsahu těchto kanálů – každý z nich je typicky 8bitový, tedy umožňuje vyjádření hodnot v rozsahu <0; 255>. To sice postačuje pro *standardní dynamický*

rozsah (SDR) reprezentovaný hodnotami $\langle 0,0; 1,0 \rangle$, ale nástroj generuje lightmapy ve *vysokém dynamickém rozsahu* (HDR), tedy s potenciálně neomezeným rozsahem $\langle 0,0; \infty \rangle$.

Nástroj proto využívá tzv. *RGBM kódování* barev. Jeho základní myšlenka spočívá v tom, že HDR barvy se nejprve normalizují vzhledem k předem zvolenému maximálnímu rozsahu. Poté se určí hodnota kanálu *M*, která odpovídá maximální složce z těchto normalizovaných RGB hodnot. Hodnota *M* zajistí, že žádná složka nepřesáhne hodnotu 1.0. Normalizované RGB složky se pak dále normalizují pomocí této hodnoty *M* tak, aby se vešly do rozsahu $\langle 0; 1 \rangle$, a výsledné hodnoty se uloží jako čtyřkanálová barva RGBM. Při dekódování je pak původní HDR barva rekonstruována vynásobením RGB složek hodnotou *M* a následně i maximálním rozsahem. Tímto způsobem lze zajistit relativně široký rozsah barev při zachování kompatibility s běžnými 8bitovými formáty, jako je PNG. Bližší popisem RGBM kódování se zabývá Karis (2009).

Pokud jsou však HDR barvy reprezentovány v lineárním barevném prostoru, zůstává kvantizační chyba i při použití RGBM kódování poměrně výrazná. V praxi se to projevuje jako viditelný vizuální artefakt v podobě tzv. *bandingu* – tedy vytváření jednobarevných pruhů v oblastech, kde by měl být přechod světla plynulý. Aby se tomuto jevu předešlo, nástroj před použitím RGBM kódování převádí barvy z lineárního barevného prostoru do gama prostoru se standardní korekcí $\gamma=2,2$. Tento převod odráží přirozenou logaritmickou závislost lidského vnímání jasu, díky čemuž se více číselných hodnot soustředí do tmavších oblastí, kde je oko na rozdíly nejcitlivější. Výsledkem je redistribuce kvantizační chyby směrem k oblastem, na které je oko méně citlivé, což zajišťuje plynulejší přechody i při použití omezené bitové hloubky.

Nástroj tak při RGBM kódování využívá maximálního rozsahu $\langle 0; 5 \rangle$ v gama prostoru a $\langle 0; 5^{2,2} \rangle$, tj. $\langle 0; 34,49 \rangle$, v lineárním prostoru. Kódování RGBM o stejném rozsahu používá k ukládání HDR lightmap mimo jiné i herní engine Unity (Unity Technologies 2019).

Používá-li formát osvětlení v lightmapě více než jeden koeficient na texel, ukládají se koeficienty zvlášť do samostatných souborů. To znamená, že v případě lightmap ve formátu SH třetího řádu, je vytvořeno celkem devět souborů. O čtení a zápis souboru PNG se stará knihovna *stb* (Barrett nedatováno).

6 Vykreslovací 3D engine

Lightmapping je uživateli v nástroji zpřístupněn pomocí uživatelského rozhraní společně s vizualizací scény. Uživatel má tak možnost si scénu zobrazit, pohybovat se v ní, vytvořit světelnou konfiguraci umístěním světelných zdrojů a v neposlední řadě generovat lightmapy s možností sledování průběhu celého procesu.

Za tímto účelem byl navržen a implementován jednoduchý vykreslovací 3D engine využívající hardwarové akcelerace prostřednictvím GPU. Engine je postaven na grafickém API *OpenGL 4.6*. Jeho architektura byla přizpůsobena výhradně potřebám nástroje. Hlavním cílem bylo umožnit vizualizaci scény v reálném čase, nikoliv dosáhnout maximální grafické kvality nebo výkonu. Engine proto nebyl výrazně optimalizován a ani nevyužívá pokročilejších zobrazovacích technik.

Výjimku představuje implementace techniky *Weighted Blended Order-Independent Transparency* (McGuire a Bavoil 2013), která byla použita pro vykreslování průhledných materiálů v reálném čase¹⁵.

Dynamické osvětlení je pak implementováno formou tzv. *forward shadingu*, kdy se při vykreslování každého geometrického objektu v rámci scény přímo vyhodnocuje vliv všech světelných zdrojů při průchodu každého vykreslovaného fragmentu. Tento výpočet probíhá bez ohledu na to, zda daný světelný zdroj skutečně působí na daný objekt, nebo zda daný fragment nebude následně přepsán jiným fragmentem objektu, který se nachází blíže kameře. Tento postup, ačkoliv snadný na implementaci, je výpočetně náročný a při větším množství světelných zdrojů a složitější geometrii se stává neefektivním, což může vést k výraznému poklesu snímkové frekvence.

6.1 Zobrazovací módy

Nástroj byl navržen s důrazem na interaktivitu a edukativní charakter, aby uživateli umožnil lépe porozumět principům lightmappingu a osvětlení v počítačové grafice obecně.

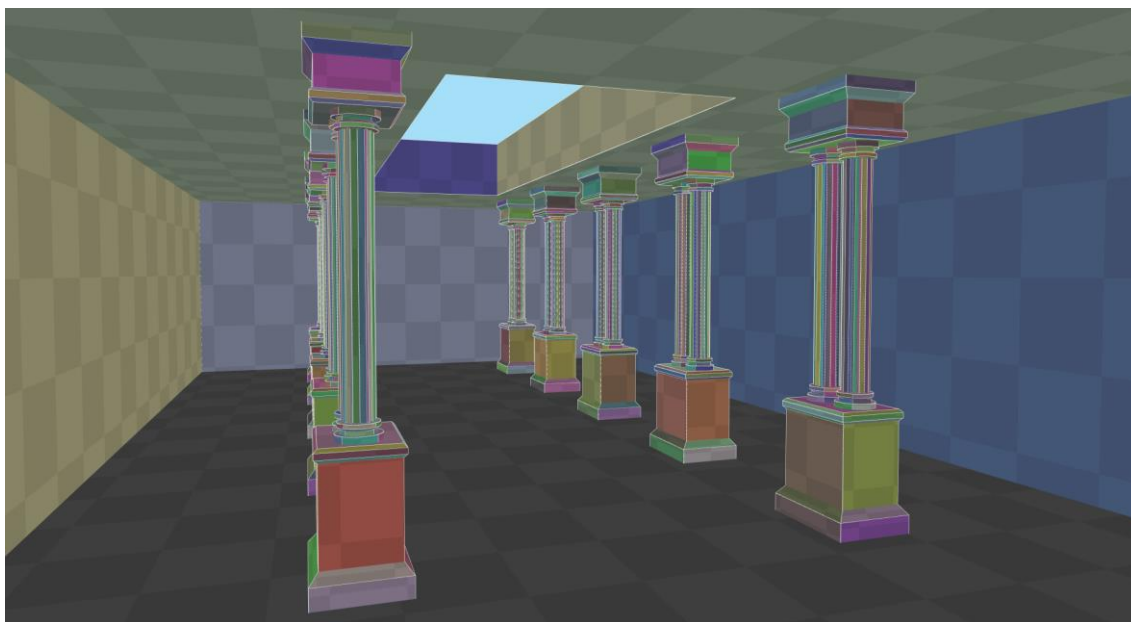
Klíčovou pro porozumění těmto principům je vizuální zpětná vazba. Z tohoto důvodu nástroj poskytuje celkem tři různé zobrazovací módy, které jsou při lightmappingu užitečnými ať už pro účely edukativní či praktické.

První z nich je základní stínovaný mód, jenž scénu zobrazuje v podmínkách produkčního prostředí – tedy tak, jak by mohla ideálně vypadat například ve videohře. Tento mód je navíc do značné míry konfigurovatelný a umožňuje selektivně aktivovat či deaktivovat konkrétní prvky vykreslování, jako je použití albeda, normálových map, lightmap, bikubického filtrování či dynamického výpočtu osvětlení.

Další zobrazovací módy slouží výhradně pro ladění a diagnostiku lightmapového atlasu. Jedním z nich je mód, který zobrazuje příslušnost povrchů k určitému chartu v lightmapovém atlasu. Součástí zobrazení je i vykreslení jednotlivých texelů ve formě šachovnicového vzoru, který

¹⁵ Tradiční přístup k vykreslování průhlednosti v reálném čase vyžaduje, aby byly trojúhelníky vykresleny ve správném pořadí podle hloubky, což obvykle znamená nutnost jejich předchozího třídění. Naproti tomu technika *Weighted Blended Order-Independent Transparency* umožňuje dosažení vizuálně uspokojivé průhlednosti i bez potřeby explicitního třídění, čímž zjednodušuje implementaci a zároveň si zachovává výpočetní efektivitu.

umožňuje diagnostikovat rozlišení atlasu. Nástroj pak zároveň v libovolném módu umožňuje zobrazit i švy chartů, tedy hran, v nichž dochází dělení částí povrchu do chartů. V těchto místech pak lze vyhledávat texely, kam bude třeba dilatovat osvětlení. Podobné zobrazení v praxi využívá například herní engine *Frostbite* (O'Donnell 2018, s. 11).



Obr. 30: Vizualizace příslušnosti povrchů k jednotlivým chartům v rámci lightmapového atlasu.

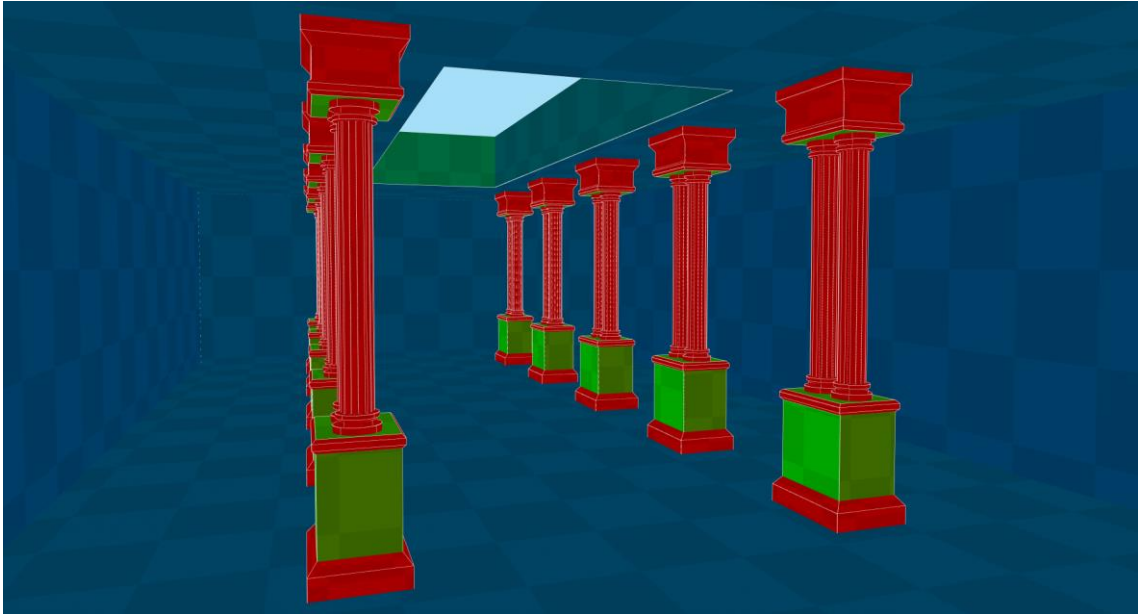
Zobrazení švů chartů je aktivní.

Zdroj: vlastní zpracování

Druhý z diagnostických módů poskytuje zobrazení hustoty texelů v lightmapovém atlasu. Uživatel tak může snadno odhalit, které části scény mají příliš nízké rozlišení, což může vést k nepřesnému osvětlení při jeho předpočítání do lightmapy. Naopak lze identifikovat i místa, kde je texelů zbytečně mnoho a dochází k plýtvání pamětí. Tento mód pomáhá najít rovnováhu mezi kvalitou a paměťovou efektivitou při rozvržení lightmapového atlasu. S vyšším rozlišením atlasu pak v druhé řadě roste i výpočetní náročnost při generování lightmap.

Uživatel má možnost zvolit rozpětí hustoty texelů prostřednictvím tří mezních hodnot, které definují rozlišení považované za nedostatečné (modře), optimální (zeleně) a nadměrné (červeně). Použitý fragment shader tato rozmezí využívá při vykreslování scény tak, že na základě poměru velikosti texelu vůči ploše, kterou texel pokrývá, přiřazuje jednotlivým oblastem odpovídající barvu.

Toto zobrazení bylo do značné míry inspirováno obdobnou funkcionalitou v herním enginu *Unreal* (Epic Games nedatováno). Za předpokladu, že jednotkou vzdálenosti ve scéně je 1 metr, považuje Unreal Engine ve výchozím nastavení za optimální hustotu texelů hodnotu 20 cm/texel (Epic Games nedatováno). O'Donnell (2018, s. 11) uvádí, že společnost Electronic Arts při vývoji videohry *Star Wars Battlefront* (vydáno 2015) cílila na hustotu 50 cm/texel a při vývoji *Star Wars Battlefront II* (vydáno 2017) pak na 25 cm/texel, přičemž generované lightmapy ukládaly pouze nepřímé osvětlení (O'Donnell 2018, s. 9).



Obr. 31: Vizualizace hustoty texelů v rámci lightmapového atlasu. Barevné označení povrchů vychází z poměru velikosti texelu vůči ploše, kterou pokrývá – modře nedostatečné, zeleně optimální a červeně nadměrné rozlišení. Zobrazení švů chartů je aktivní.

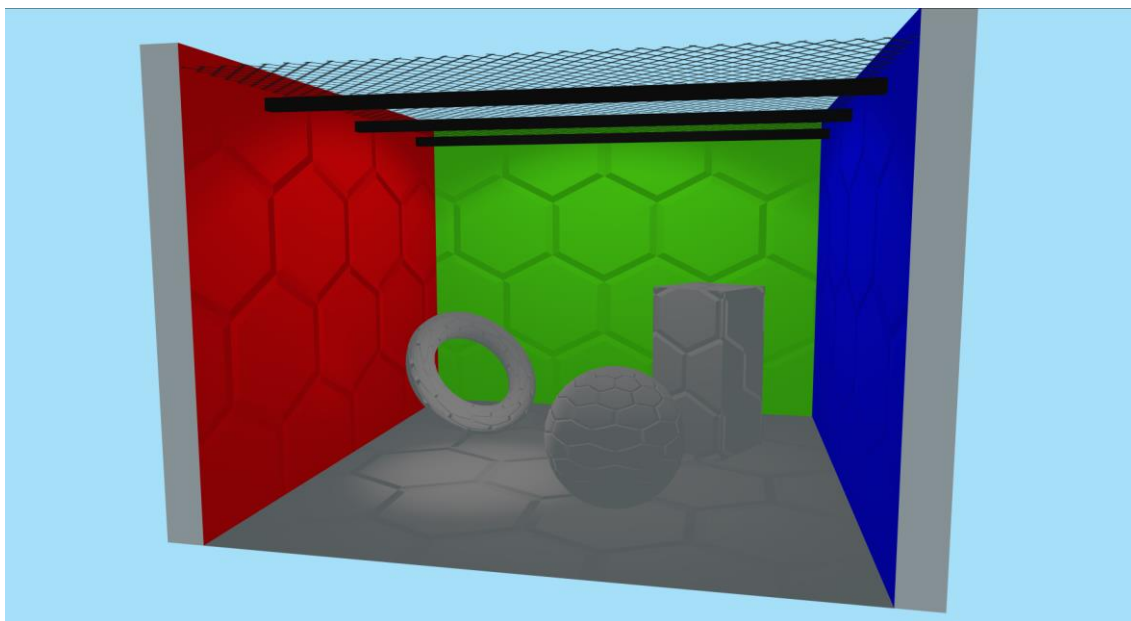
Zdroj: vlastní zpracování

7 Výsledky

Pro demonstraci výsledků a schopností nástroje byla zvolena scéna, která umožňuje vhodně prezentovat různé jevy osvětlení a přednosti jednotlivých formátů lightmap. Scéna byla inspirována projektem *The Baking Lab*, v kterém autoři Neubelt a Pettineo použili podobnou scénu při testování různých metod ukládání předpočítaného osvětlení (Pettineo 2016 b).

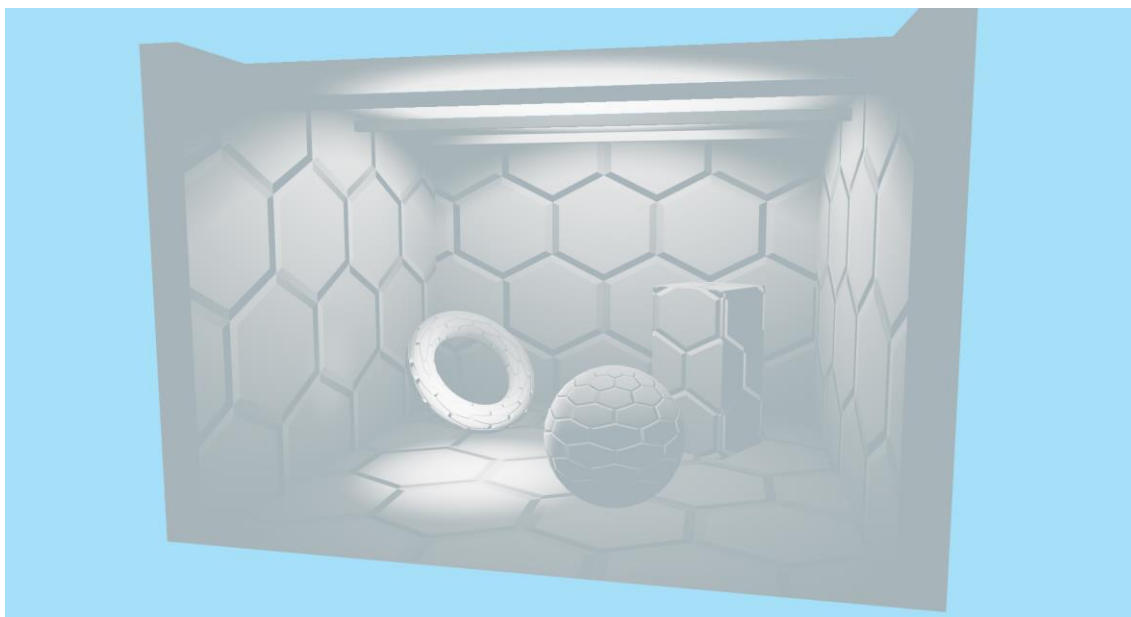
Scéna je osvětlena dvěma světelnými zdroji – jedním reflektorovým světlem nacházejícím se nad scénou a jedním bodovým světlem, nacházejícím se uvnitř scény. Při vzorkování lightmap je využito bikubického filtrování. Pro výpočet lightmap bylo zvolena konfigurace s počtem vzorků = 64 a maximálním počtem odrazů = 4.

Různé barvy stěn ve scéně umožňují vhodně demonstrovat nepřímé osvětlení. Pomocí malých objektů ve scéně lze demonstrovat stíny, které vznikají v důsledku blokování světla povrchem objektů. Pro ukázkou průsvitnosti povrchů je využito povrchu, který je pokryt částečně průhlednou texturou – pletivem, které je umístěno na stropu scény. Schopnosti směrových lightmap jsou dále ilustrovány pomocí normálových map aplikovaných na povrchy stěn a objektů.



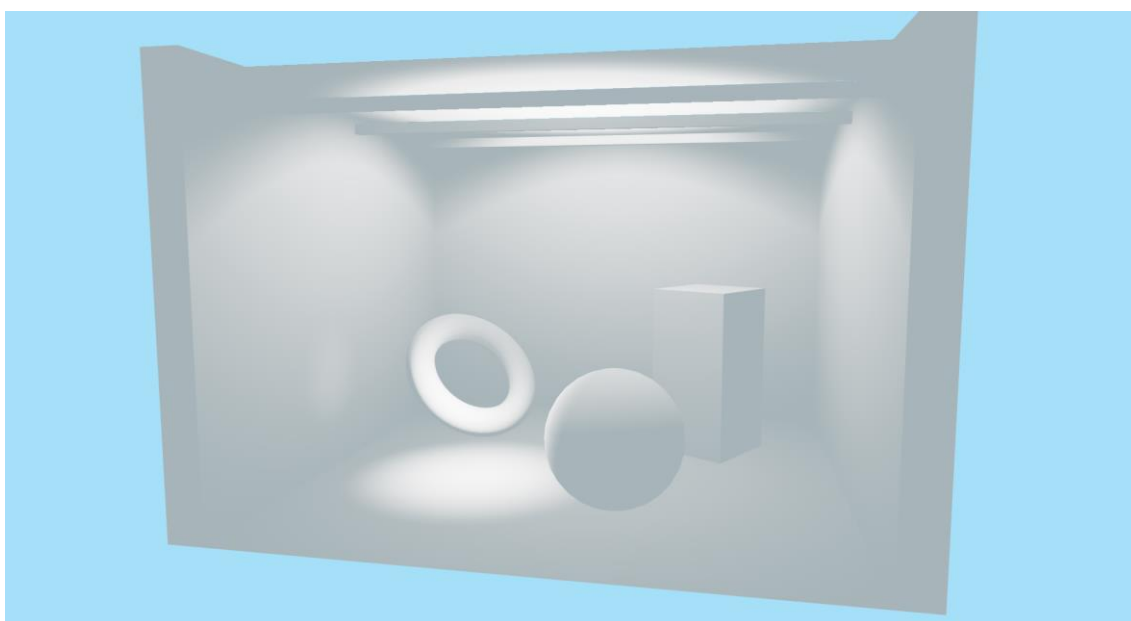
Obr. 32: Scéna osvětlená pomocí dynamicky počítaného přímého osvětlení. Povrch je pokryt barvou albeda, při výpočtu osvětlení je využíváno normály z normálové mapy.

Zdroj: vlastní zpracování



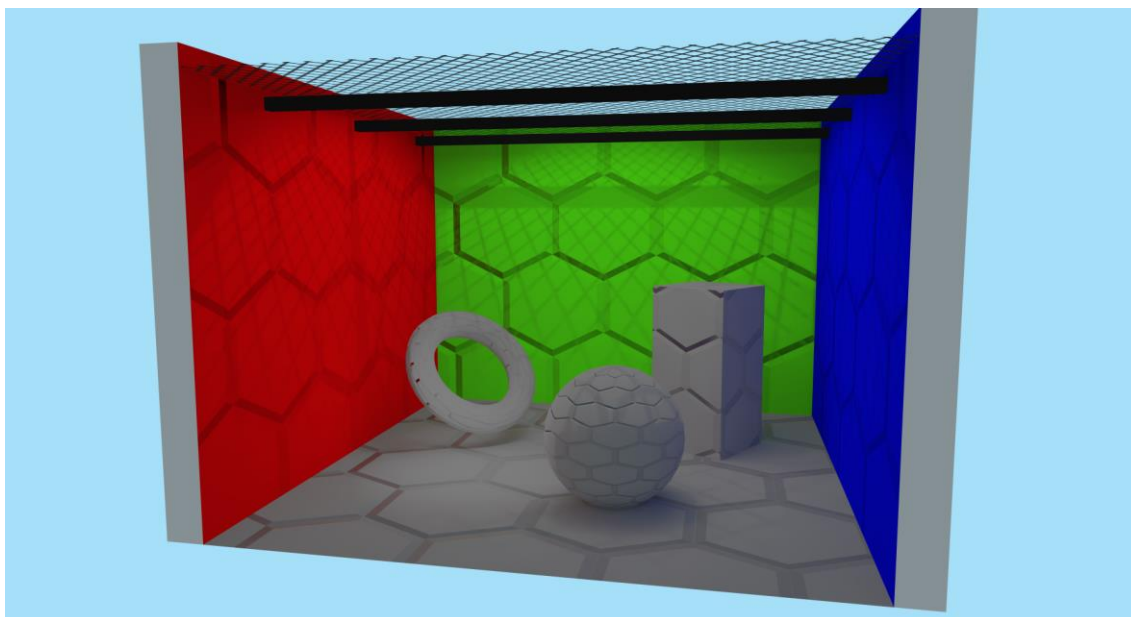
Obr. 33: Scéna osvětlená pomocí dynamicky počítaného přímého osvětlení. Na povrchu je zobrazeno pouze osvětlení, při výpočtu osvětlení je využíváno normály z normálové mapy.

Zdroj: vlastní zpracování



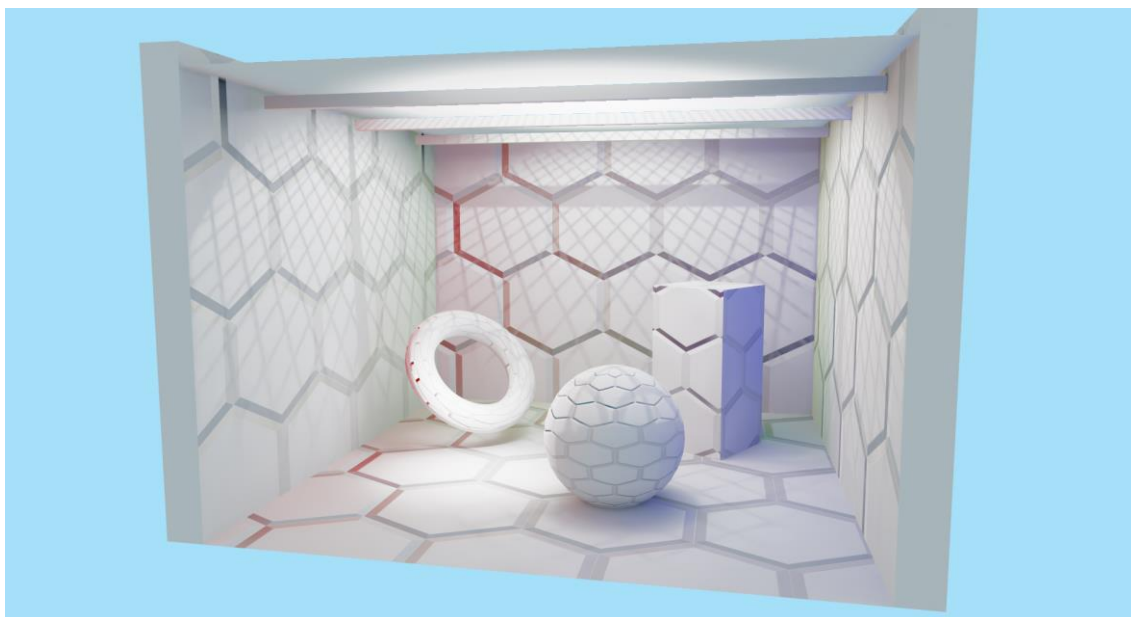
Obr. 34: Scéna osvětlená pomocí dynamicky počítaného přímého osvětlení. Na povrchu je zobrazeno pouze osvětlení, při výpočtu osvětlení je využíváno povrchové normály.

Zdroj: vlastní zpracování



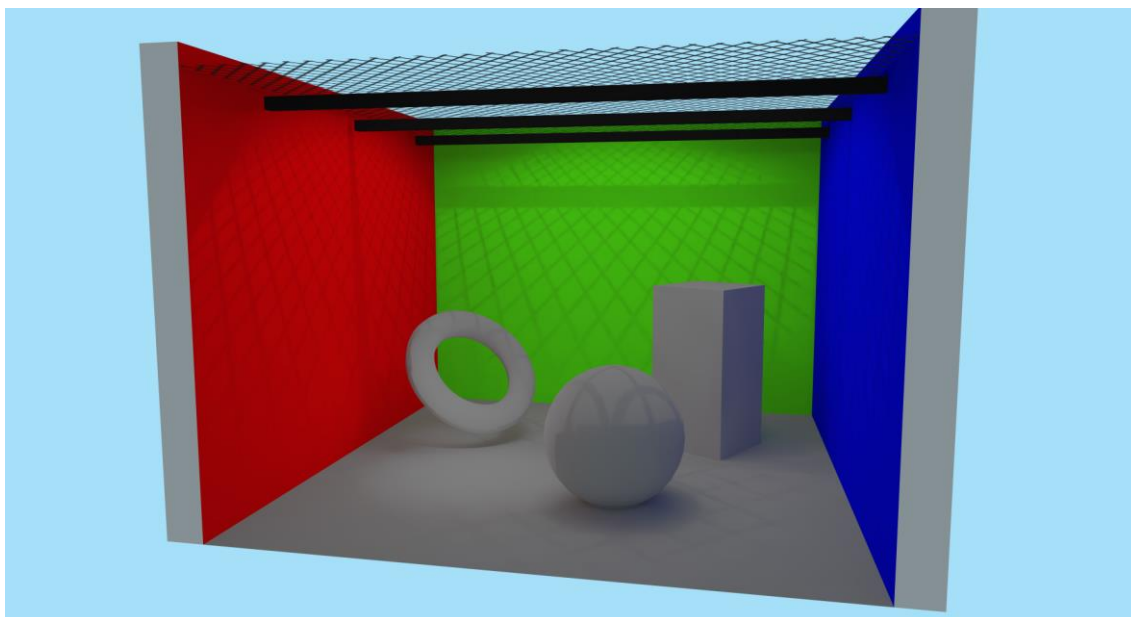
Obr. 35: Scéna osvětlená pomocí předpočítaného globálního osvětlení ze směrové lightmapy ve formátu SH třetího řádu. Povrch je pokryt barvou albeda, při rekonstrukci osvětlení z lightmapy je využíváno normály z normálové mapy.

Zdroj: vlastní zpracování



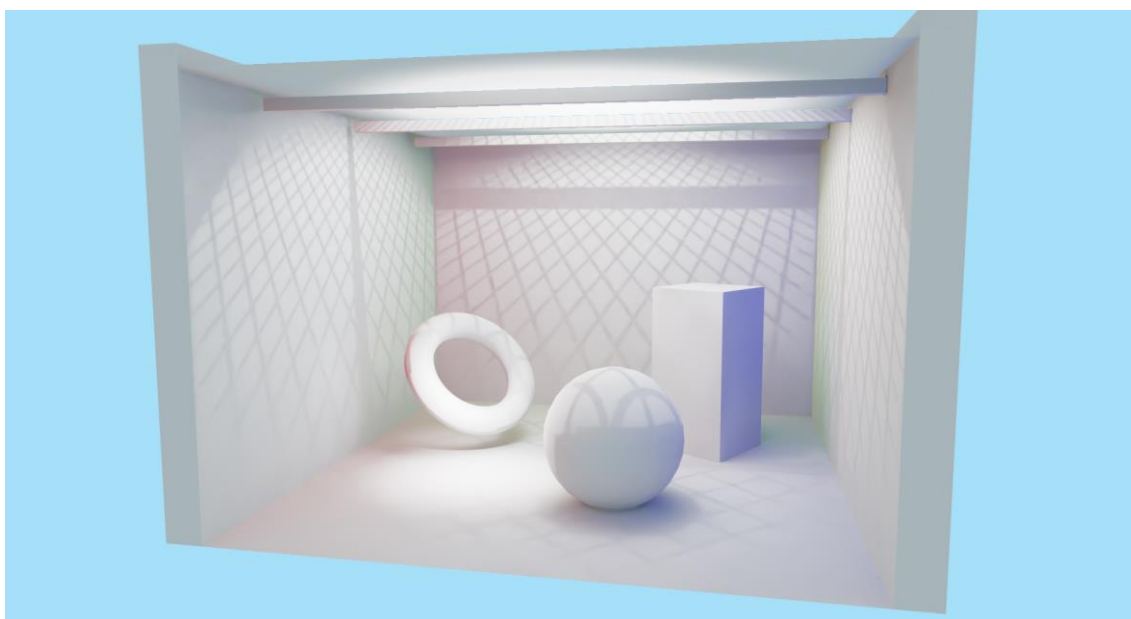
Obr. 36: Scéna osvětlená pomocí předpočítaného globálního osvětlení ze směrové lightmapy ve formátu SH třetího řádu. Na povrchu je zobrazeno pouze osvětlení, při rekonstrukci osvětlení z lightmapy je využíváno normály z normálové mapy.

Zdroj: vlastní zpracování



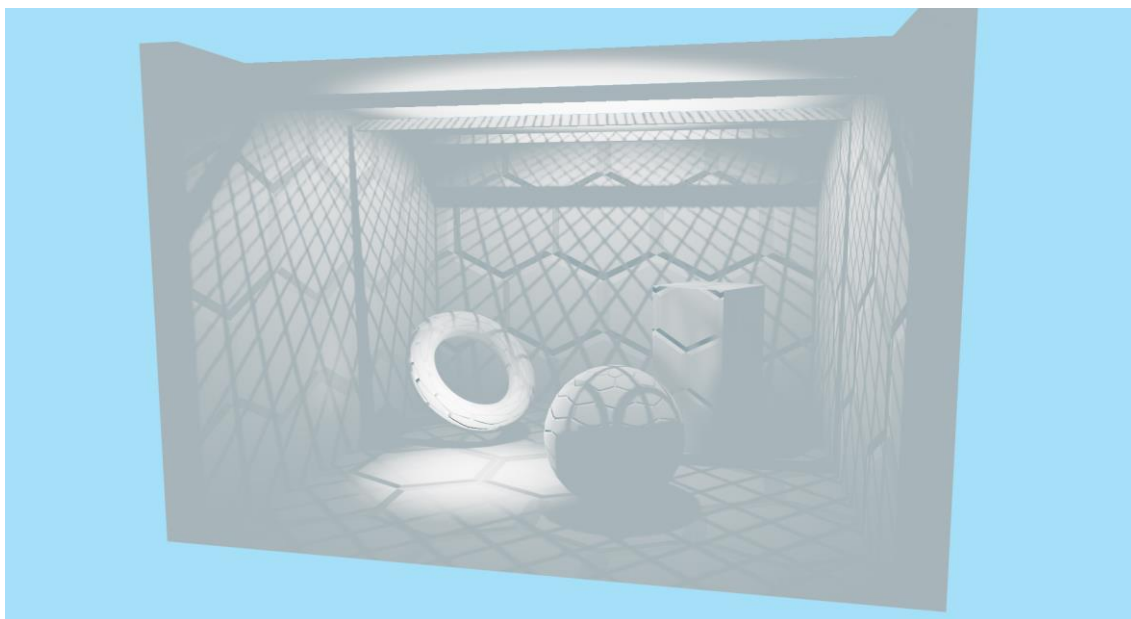
Obr. 37: Scéna osvětlená pomocí předpočítaného globálního osvětlení z nesměrové lightmapy. Povrch je pokryt barvou albeda, při rekonstrukci osvětlení z lightmapy je implicitně využíváno povrchové normály.

Zdroj: vlastní zpracování



Obr. 38: Scéna osvětlená pomocí předpočítaného globálního osvětlení z nesměrové lightmapy. Na povrchu je zobrazeno pouze osvětlení, při rekonstrukci osvětlení z lightmapy je implicitně využíváno povrchové normály.

Zdroj: vlastní zpracování



Obr. 39: Scéna osvětlená pomocí předpočítaného přímého osvětlení ze směrové lightmapy ve formátu SH třetího řádu. Na povrchu je zobrazeno pouze osvětlení, při rekonstrukci osvětlení z lightmapy je využíváno normály z normálové mapy.

Zdroj: vlastní zpracování

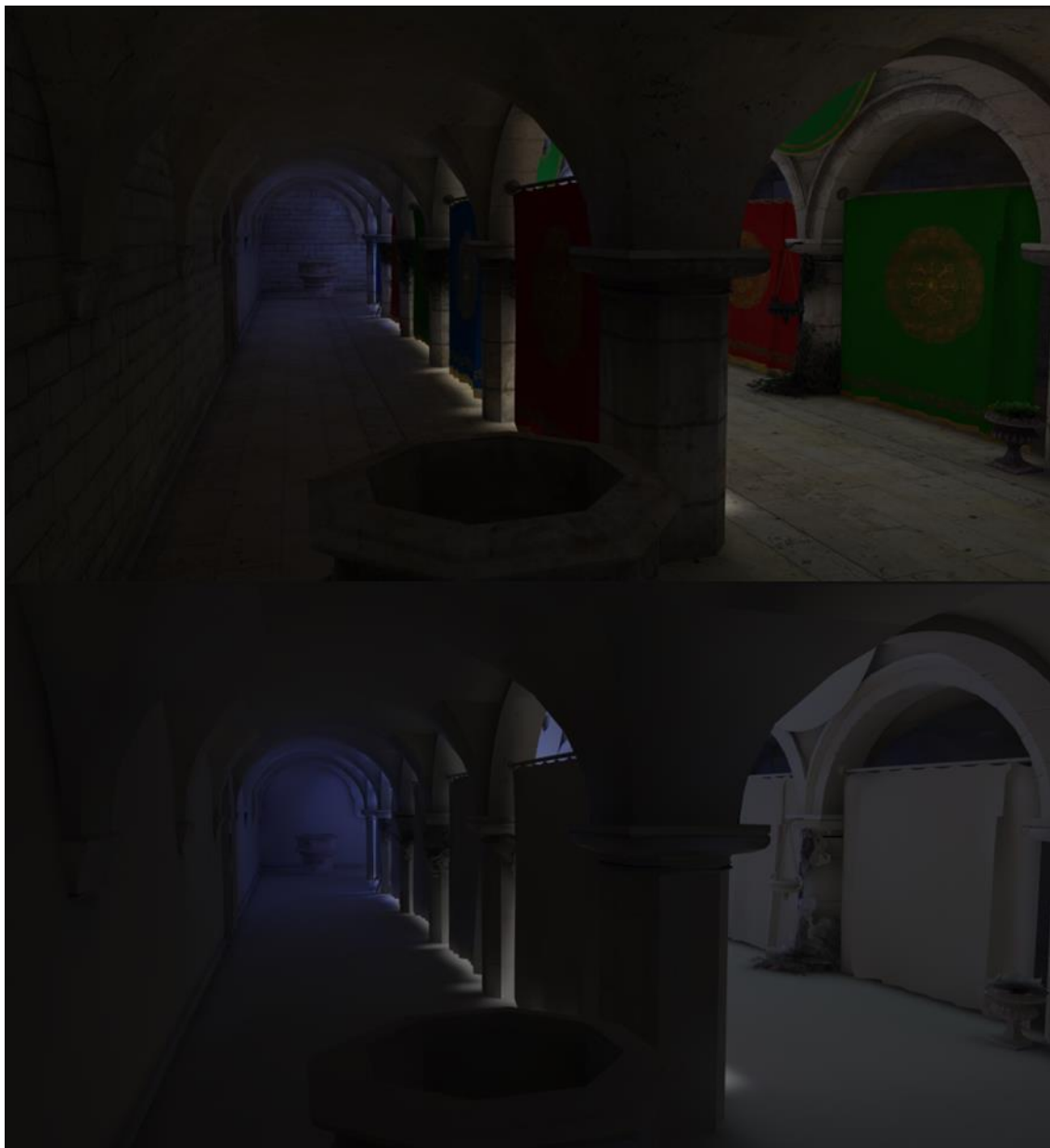
Pro reprezentativní prezentaci výsledků byla dále zvolena scéna Sponza. Původní 3D model vznikl v roce 2002, jehož autorem je Marko Dabrovic (The Khronos Group nedatováno b). V této práci je však využita novější varianta vytvořená Frankem Meinlem, která je dostupná jako součást vzorových 3D modelů v oficiálním repozitáři formátu glTF (The Khronos Group nedatováno b). Tato scéna patří mezi standardní testovací prostředí používaná v oblasti počítačové grafiky. Nabízí komplexní architektonickou strukturu s množstvím detailů a otevřených i uzavřených prostor, což tvoří vhodný základ pro testování kvality globálního osvětlení jakož i schopností lightmappingu. Scéna Sponza rovněž svým složitějším prostředím lépe odpovídá praktickému využití, jaké se běžně vyskytuje ve videohrách nebo architektonických vizualizacích.

Scéna je osvětlena dvěma zdroji světla – směrovým světlem s bílou barvou představujícím slunce a bodovým světlem s modrou barvou uprostřed scény. Scéna využívá jediného lightmapového atlasu o velikosti 2048x2048 texelů. Pro výpočet lightmapy bylo zvolena konfigurace s počtem vzorků = 128 a maximálním počtem odrazů = 4.



Obr. 40: Atrium scény Sponza (autor: Frank Meinl) osvětlené pomocí předpočítaného globálního osvětlení ze směrové lightmapy ve formátu SH třetího řádu. Horní část zobrazuje scénu s plnými materiály a texturami, zatímco spodní část zobrazuje pouze osvětlení.

Zdroj: vlastní zpracování



Obr. 41: Postranní část atria scény Sponza (autor: Frank Meinl) osvětlené pomocí předpočítaného globálního osvětlení ze směrové lightmapy ve formátu SH třetího řádu. Horní část zobrazuje scénu s plnými materiály a texturami, zatímco spodní část zobrazuje pouze osvětlení.

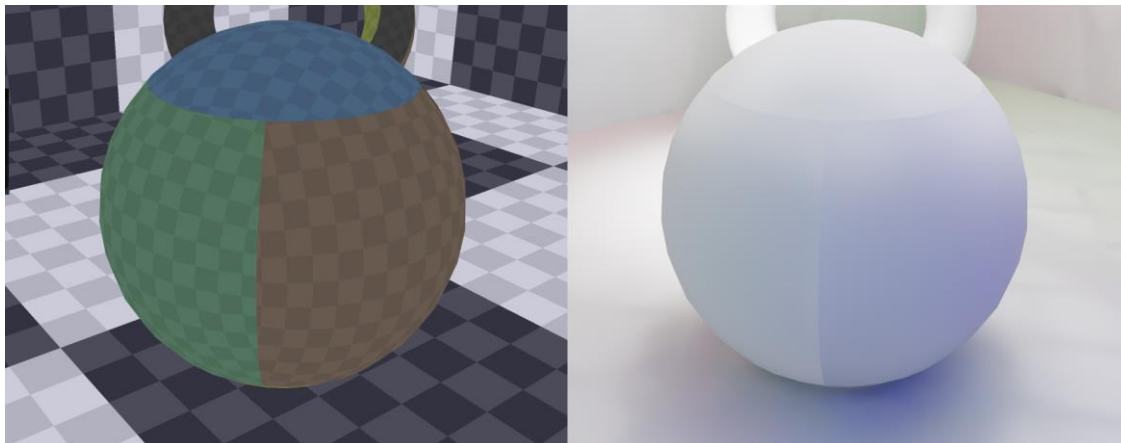
Zdroj: vlastní zpracování

7.1 Známá omezení

Ačkoliv nástroj dokáže generovat vizuálně uspokojivé výsledky, jejich dosažení může vyžadovat opakované pokusy a ladění parametrů. Při použití nástroje by proto uživatel měl rozumět alespoň základním principům lightmappingu a vědět, jakým způsobem upravovat parametry generování tak, aby dosáhl vyšší kvality výsledného osvětlení. Mimo to je třeba respektovat i omezení, která jsou spojena s použitím nástroje.

Prvním, již dříve zmíněným omezením, je problém s únikem světla (viz 4.2.1). Ovšem za zásadní omezení považují zejména absenci algoritmu pro vyhlazení ostrých přechodů v osvětlení

v místech švů, kde dochází ke styku hran mezi dvěma charty. V těchto oblastech mohou vznikat viditelné nespojitosti, jelikož vypočtené hodnoty osvětlení na sousedících částech povrchu se liší. Výsledné osvětlení pak působí nepřírozně a narušuje celkový vizuální dojem ze scény. Tento jev je nejvýraznější při nízkém rozlišení lightmapy a s vyšším rozlišením se vytrácí. Při pečlivém pozorování však bude bez explicitního vyhlazení viditelný vždy.



Obr. 42: Nespojitosti osvětlení ve švech chartů. Levá část zobrazuje rozdělení povrchu do jednotlivých chartů (barevně odlišené oblasti), zatímco pravá část ukazuje výsledné osvětlení po vygenerování lightmapy. Ve švech mezi charty jsou patrné ostré přechody v osvětlení.

Zdroj: vlastní zpracování

Závěr

Cílem práce bylo seznámit se s možnostmi předpočítaného osvětlení se zaměřením na lightmapy a zároveň navrhnout a implementovat nástroj, který umožní jejich generování pro libovolnou uživatelem definovanou scénu.

Teoretická část práce se zabývá problematikou globálního osvětlení v počítačové grafice a popisuje metody jeho předpočítání pro dosažení realistického osvětlení scén vykreslovaných v reálném čase. Úvodní kapitola poskytuje přehled základních principů světla a jeho chování v počítačové grafice. Následně je popsáno globální osvětlení a objasněna motivace pro jeho předpočítání.

Následující kapitoly se soustředí na konkrétní metody reprezentace předpočítaného osvětlení a popis veličin, které se do těchto reprezentací ukládají. Jako stěžejní metoda této práce jsou podrobně popsány lightmapy coby zástupce předpočítaného osvětlení na úrovni povrchu. Zároveň jsou ale uvedeny i alternativní přístupy zaměřené na předpočítání osvětlení na úrovni prostoru. Jak v případě reprezentací, tak v případě veličin a jejich formátu jsou uvedeny aspekty, jež ovlivňují volbu vhodné metody pro danou aplikaci, společně s jejich výhodami a omezeními.

Vzhledem k tomu, že se práce dotýká řady pokročilých témat – od fyzikálního modelování světla přes vzorkovací strategie až po grafiku a 3D vykreslování – nebylo možné jít do úplné hloubky ve všech oblastech. Důraz byl proto kladen na funkční propojení těchto komponent do uceleného nástroje a porozumění základním principům, na kterých jsou postaveny.

Praktická část práce se pak zaměřuje na návrh, implementaci a testování softwarového nástroje, který slouží pro demonstraci procesu generování lightmap. Pro výpočet globálního osvětlení nástroj využívá metody path tracingu, jenž byl zvolen pro svou relativní jednoduchost implementace, možnost snadné paralelizace a schopnost dosáhnout vysoké kvality osvětlení.

V rámci návrhu softwarového řešení byla navržena architektura aplikace, která umožňuje automatizovat celý proces generování výsledné lightmapy, včetně tvorby unikátní UV parametrizace povrchu scény prostřednictvím lightmapového atlasu. Součástí návrhu byla také volba vhodné reprezentace výstupních dat scény, přičemž pozornost je věnována zejména kódování lightmap v HDR pomocí RGBM kódování.

Aplikace navíc poskytuje vizuální editor, který umožňuje uživateli volný pohyb ve scéně pomocí kamery, vytvářet a upravovat světelné zdroje či sledovat probíhající generování lightmap. Mimo to poskytuje i další zobrazovací módy, jež umožňují ladění a diagnostiku lightmapového atlasu.

Návrh řešení vychází z obecných postupů a osvědčených doporučení pro návrh nástrojů tohoto typu. Přestože v některých částech využívá vlastní jednoduché akcelerační struktury a optimalizační postupy, tyto prvky nepředstavují zásadní inovaci a jejich implementace nevyžaduje složité řešení.

Z povahy path tracingu vyplývá, že pro dosažení kvalitních výsledků je nutné vypočítat velké množství vzorků, než se dosáhne vizuálně uspokojivého výstupu. Při nízkém počtu vzorků bývá výstup zatížen šumem. Z tohoto důvodu nástroj podporuje progresivní výpočet, který umožňuje uživateli sledovat postupné zpřesňování výsledné lightmapy v reálném čase, přičemž má možnost výpočet kdykoli zastavit, považuje-li dosažený výsledek za dostatečný. Další výhodou progresivního přístupu je, že uživatel nemusí čekat na dokončení výpočtu lightmap, ale může se

během něj věnovat jiné práci v editoru. Přestože má nástroj převážně demonstrační charakter a rozsah dostupných funkcí je velmi omezený, tato vlastnost zůstává z obecného hlediska velmi užitečná, neboť v reálných aplikacích umožňuje efektivnější využití času uživatele.

Pro redukci šumu ve výsledné lightmapě je po dokončení výpočtu aplikován denoisingový filtr, který výrazně zlepšuje kvalitu osvětlení a zároveň umožňuje dosáhnout uspokojivých výsledků s nižším počtem vzorků. Denoisingový filtr lze zároveň volitelně aktivovat i kdykoliv během probíhajícího výpočtu.

Stanovený cíl práce se podařilo splnit. Jejím výsledkem je v první řadě samotný text, který poskytuje ucelený přehled o metodě lightmappingu a souvisejících technických a implementačních otázkách spojených s jeho realizací. V řadě druhé pak jde o funkční softwarový nástroj, který demonstruje proces generování lightmap se všemi náležitostmi nezbytnými pro jejich využití v praxi. Výstupem nástroje jsou pak lightmapy, jež dosahují přesvědčivé vizuální kvality, ovšem s určitými nedostatky. Mimo to nástroj poskytuje i nadstandardní funkce, které zvyšují uživatelskou přívětivost.

Nástroj byl navržen jako samostatná aplikace, která pracuje se svou vlastní reprezentací scény, jakož i vlastními formáty dat. Integrace jeho výstupů do produkčního řešení by tak vyžadovala podřízení se jeho reprezentaci scény, zejména modelům světelných zdrojů a materiálům.

Jako takový tedy neumožňuje přímou integraci do externího produkčního řešení. Nicméně samotné generování lightmap bylo navrženo tak, aby jej bylo možné oddělit do samostatného modulu, který by po určitých úpravách bylo možné přenést a přizpůsobit potřebám jiných aplikací.

Nástroj umožňuje generovat jak základní nesměrové lightmapy, tak směrové lightmapy, jež lze za běhu využít společně s technikou normal mappingu. V současné době je však nástrojem pro reprezentaci směrových lightmap podporován pouze formát SH třetího řádu. Díky modulárnímu návrhu rozhraní, které využívá tzv. akumulátory, by však nemělo být problematické nástroj v budoucnu rozšířit i o další formáty lightmap.

Navzdory dosaženým výsledkům však nástroj doprovázejí jistá omezení. Hlavním z nich je absence algoritmu pro vyhlazení ostrých přechodů v osvětlení v místech švů. Druhým omezením je výskyt artefaktů ve formě úniku světla na hranách geometrických objektů, které se nacházejí na rozhraní dvou odlišně osvětlených prostředí. Před nasazením nástroje do produkčního prostředí by proto bylo vhodné tyto nedostatky nejprve odstranit.

Jako další omezení může být chápáno, že vygenerované lightmapy lze použít pouze pro předpočítání difúzního osvětlení a nelze jej tak využít pro předpočítání pokročilejších BRDF se spekulární složkou. Nástroj dále podporuje pouze bezrozměrné světelné zdroje, což znamená, že neumožňuje reprezentovat povrchy, které emitují světlo ani jiné plošné světelné zdroje.

Výpočet lightmap by bylo dále možné akcelarovat pomocí GPU, jež by zvýšilo výpočetní efektivitu. Taková změna by však vyžadovala zásadní přepracování aplikace, neboť by vyžadovala přepsání algoritmů a datových struktur do podoby vhodné pro běh na GPU.

Mezi další návrhy na zefektivnění výpočtu řadím použití adaptivního vzorkování nepřímého osvětlení řízeného na základě rozptylu hodnot z předchozích vzorků. To znamená, že pro texely

s vyšším rozptylem hodnot nepřímého osvětlení by bylo alokováno více dalších vzorků, zatímco u texelů s nižším rozptylem by bylo možné vzorkování ukončit dříve.

V neposlední řadě by šlo path tracing zefektivnit tím, že v případě dopadu paprsku na objekt, který je již pokryt lightmapou, by šlo použít hodnotu nepřímého osvětlení vzorkované přímo z ní. Nicméně takto získaná hodnota nemusí být tak přesná jako výsledek plného výpočtu pomocí path tracingu.

Samotný algoritmus implementovaného path tracingu je však obecně velmi jednoduchý a do budoucna by si zasloužil rozšíření i o pokročilejší fyzikální jevy či optimalizační kroky.

Přestože je nástroj primárně zaměřen na demonstrační účely a není určen pro přímé nasazení v produkčních aplikacích, při respektování jeho omezení by bylo možné jej použít ve videohrách, které necílí na vysoce realistické výsledky a vystačí si i s předpočítaným difúzním globálním osvětlením.

S ohledem na dosažené výsledky a navržené možnosti dalšího rozšíření lze považovat tuto práci za úspěšnou. Vytvořený nástroj nejen splňuje své demonstrační poslání, ale zároveň nabízí solidní základ pro další vývoj směrem k plnohodnotnému produkčnímu řešení.

Seznam použité literatury

- AKENINE-MÖLLER, Tomas; HAINES, Eric; HOFFMAN, Naty; PESCE, Angelo; IWANICKI, Michael et al. *Real-time rendering*. Fourth edition. Boca Raton: CRC Press/Taylor & Francis Group, [2018]. ISBN 978-1-138-62700-0.
- APELER, Sean. *Fastgltf*. Online. GitHub. Dostupné z: <https://github.com/spnda/fastgltf>. [cit. 2025-04-10].
- APERS, Diede; EDBLOM, Petter; ROUSIERS, Charles de a HILLAIRES, Sébastien. Interactive Light Map and Irradiance Volume Preview in Frostbite. Online. In: HAINES, Eric a AKENINE-MÖLLER, Tomas (ed.). *Ray Tracing Gems*. Berkeley, CA: Apress, 2019, s. 377-407. ISBN 978-1-4842-4426-5. Dostupné z: https://doi.org/10.1007/978-1-4842-4427-2_23. [cit. 2025-03-29].
- BAČÍK, Michal. *Insanity3D*. Online. GitHub. Dostupné z: <https://github.com/mice777/Insanity3D/blob/49dc70130f786439fb0e4f91b75b6b686a134760/Source/I3D2/Lmap.cpp>. [cit. 2025-04-13].
- BANCILA, Marius. *Stduuid*. Online. GitHub. Dostupné z: <https://github.com/mariusbancila/stduuid>. [cit. 2025-04-10].
- BARRETT, Sean. *Stb*. Online. GitHub. Dostupné z: <https://github.com/nothings/stb>. [cit. 2025-04-10].
- BLENDER FOUNDATION. *Light Objects*. Online. Blender 4.3 Manual. 2025, 1.10.2025. Dostupné z: https://docs.blender.org/manual/en/4.3/render/lights/light_object.html#spot-light. [cit. 2025-03-13].
- BOKSANSKY, Jakub. *Crash Course in BRDF Implementation*. Online. 2021. Dostupné z: <https://boksajak.github.io/blog/BRDF>. [cit. 2024-11-16].
- BOKSANSKY, Jakub a MARRS, Adam. The Reference Path Tracer. Online. In: MARRS, Adam; SHIRLEY, Peter a WALD, Ingo (ed.). *Ray Tracing Gems II*. Berkeley, CA: Apress, 2021, s. 161-187. ISBN 978-1-4842-7184-1. Dostupné z: https://doi.org/10.1007/978-1-4842-7185-8_14. [cit. 2025-02-19].
- BRITANNICA KIDS. *Shadows: point versus extended sources*. Online. Britannica Kids. Nedatováno. Dostupné z: <https://kids.britannica.com/students/assembly/view/229835#>. [cit. 2025-01-10].
- BUSH, Jeff. *Quake Lightmaps*. Online. Project log. 11.6.2015. Dostupné z: <https://jbush001.github.io/2015/06/11/quake-lightmaps.html>. [cit. 2025-04-12].
- CORNUT, Omar. *Dear ImGui*. Online. GitHub. Dostupné z: <https://github.com/ocornut/imgui>. [cit. 2025-04-10].
- CRANLEY, R. a PATTERSON, T. N. L. Randomization of Number Theoretic Methods for Multiple Integration. Online. *SIAM Journal on Numerical Analysis*. 1976, roč. 13, č. 6, s. 904-914. ISSN 0036-1429. Dostupné z: <https://doi.org/10.1137/0713071>. [cit. 2025-04-05].

- CUPISZ, Robert. *Light Probe Interpolation Using Tetrahedral Tessellations*. Online. 2012. Dostupné z: <https://gdcvault.com/play/1015312/Light-Probe-Interpolation-Using-Tetrahedral>. [cit. 2025-01-10].
- DEBEVEC, Paul. *Light Probe Image Gallery*. Online. Paul Debevec. 1998, 9.10.2004. Dostupné z: <https://www.pauldebevec.com/Probes/>. [cit. 2025-03-29].
- DE VRIES, Joey. *PBR – Theory*. Online. LearnOpenGL. Dostupné z: <https://learnopengl.com/PBR/Theory>. [cit. 2024-11-17].
- EPIC GAMES. *Understanding Lightmapping in Unreal Engine | Unreal Engine 5.5 Documentation | Epic Developer Community*. Online. Epic Games Developer. Dostupné z: <https://dev.epicgames.com/documentation/en-us/unreal-engine/understanding-lightmapping-in-unreal-engine>. [cit. 2025-04-11].
- GLFW. *GLFW*. Online. GitHub. Dostupné z: <https://github.com/glfw/glfw>. [cit. 2025-04-10].
- GOURAUD, Henri. Continuous shading of curved surfaces. Online. In: Seminal graphics. New York, NY, USA: ACM, 1998, s. 87-93. ISBN 158113052X. Dostupné z: <https://doi.org/10.1145/280811.280979>. [cit. 2024-12-21].
- GREEN, Robin. Spherical Harmonic Lighting: The Gritty Details. Online. 2003. Dostupné z: <https://3dvar.com/Green2003Spherical.pdf>. [cit. 2025-03-02].
- GREGGER, G.; SHIRLEY, P.; HUBBARD, P.M. a GREENBERG, D.P. The irradiance volume. Online. IEEE Computer Graphics and Applications. Roč. 18, č. 2, s. 32-43. ISSN 02721716. Dostupné z: <https://doi.org/10.1109/38.656788>. [cit. 2025-01-10].
- G-TRUC CREATION. *OpenGL Mathematics (GLM)*. Online. GitHub. Dostupné z: <https://github.com/g-truc/glm>. [cit. 2025-04-10].
- HERBERTH, David. *Glad*. Online. GitHub. Dostupné z: <https://github.com/Dav1dde/glad>. [cit. 2025-04-10].
- HOOVER, John. Volumetric Global Illumination at Treyarch. Online. 2016. Dostupné z: <https://research.activision.com/publications/archives/volumetric-global-illumination-at-treyarch>. [cit. 2025-01-10].
- HOOVER, Richard. *Retro Tech: Quake 3 Light Volumes*. Online. SeaLeft Studios. 17.6.2016. Dostupné také z: <https://www.sealeftstudios.com/blog/blog20160617.php>.
- INTEL. *Intel Embree*. Online. GitHub. Nedatováno a. Dostupné z: <https://github.com/RenderKit/embree>. [cit. 2025-04-10].
- INTEL. *Intel oneAPI Threading Building Blocks*. Online. GitHub. Nedatováno b. Dostupné z: <https://github.com/uxlfoundation/oneTBB>. [cit. 2025-04-10].
- INTEL. *Intel Open Image Denoise*. Online. Nedatováno c. GitHub. Dostupné z: <https://github.com/RenderKit/oidn>. [cit. 2025-04-10].
- IWANICKI, Michal. Lighting technology of the last of us. Online. In: *ACM SIGGRAPH 2013 Talks*. New York, NY, USA: ACM, 2013, s. 1-1. ISBN 9781450323444. Dostupné z: <https://doi.org/10.1145/2504459.2504484>. [cit. 2025-04-13].

- KAJIYA, James T. The rendering equation. Online. *ACM SIGGRAPH Computer Graphics*. 1986, roč. 20, č. 4, s. 143-150. ISSN 0097-8930. Dostupné z: <https://doi.org/10.1145/15886.15902>. [cit. 2025-04-13].
- KARIS, Brian. *RGBM color encoding*. Online. Graphic Rants. 28.4.2009. Dostupné z: <https://graphicrants.blogspot.com/2009/04/rgbm-color-encoding.html>. [cit. 2025-04-10].
- KŘIVÁNEK, Jaroslav. *Počítačová grafika III – Odraz světla, BRDF*. Online. 2012. Dostupné z: <https://cgg.mff.cuni.cz/~jaroslav/teaching/2012-NPGR010/slides/NPGR010-2012%20-%2003-BRDF.pdf>. [cit. 2025-02-20].
- LAGARDE, Sébastien a DE ROUSIERS, Charles. Moving Frostbite to Physically Based Rendering 3.0. 2014. Dostupné také z: https://seblagarde.wordpress.com/wp-content/uploads/2015/07/course_notes_moving_frostbite_to_pbr_v32.pdf.
- LINDQVIST, Anders. Multiple Importance Sampling 101. Online. In: MARRS, Adam; SHIRLEY, Peter a WALD, Ingo (ed.). *Ray Tracing Gems II*. Berkeley, CA: Apress, 2021, s. 327-337. ISBN 978-1-4842-7184-1. Dostupné z: https://doi.org/10.1007/978-1-4842-7185-8_20. [cit. 2025-02-23].
- LINIETSKY, Juan; MANZUR, Ariel a komunita herního enginu Godot. *Using Lightmap global illumination*. Online. Godot Engine documentation. Dostupné z: https://docs.godotengine.org/en/4.4/tutorials/3d/global_illumination/using_lightmap_gi.html. [cit. 2025-04-10].
- LOHMANN, Niels. *JSON for Modern C++*. Online. GitHub. Dostupné z: <https://github.com/nlohmann/json>. [cit. 2025-04-10].
- MCGUIRE, Morgan a BAVOIL, Louis. Weighted Blended Order-Independent Transparency. Online. *Journal of Computer Graphics Techniques (JCGT)*. 2013, roč. 2, č. 2, s. 122–141. Dostupné z: <http://jcgt.org/published/0002/02/09/>. [cit. 2025-04-11].s
- MELMAN, Gabi. *Spdlog*. Online. GitHub. Dostupné z: <https://github.com/gabime/spdlog>. [cit. 2025-04-10].
- MITCHELL, Jason; MCTAGGART, Gary a GREEN, Chris. Shading in valve's source engine. Online. In: *ACM SIGGRAPH 2006 Courses on - SIGGRAPH '06*. New York, New York, USA: ACM Press, 2006, 129-. ISBN 1595933646. Dostupné z: <https://doi.org/10.1145/1185657.1185832>. [cit. 2025-04-13].
- NVIDIA. *Don't be conservative with Conservative Rasterization*. Online. NVIDIA Developer. Nedatováno a. Dostupné z: <https://developer.nvidia.com/content/dont-be-conservative-conservative-rasterization>. [cit. 2025-03-30].
- NVIDIA. *NVIDIA OptiX™ Ray Tracing Engine*. Online. NVIDIA Developer. Nedatováno b. Dostupné z: <https://developer.nvidia.com/rtx/ray-tracing/optix>. [cit. 2025-04-10].
- O'DONNELL, Yuriy. *Precomputed Global Illumination in Frostbite*. Online. Games Developers Conference, 2018. 2018. Dostupné z: <https://media.contentapi.ea.com/content/dam/eacom/frostbite/files/gdc2018-precomputedglobalilluminationinfrostbite.pdf>. [cit. 2024-11-02].

- PETTINEO, Matt. *SG Series Part 1: A Brief (and Incomplete) History of Baked Lighting Representations*. Online. The Danger Zone. 10.10.2016 a. Dostupné z: <https://therealmjp.github.io/posts/sg-series-part-1-a-brief-and-incomplete-history-of-baked-lighting-representations/>. [cit. 2024-11-09].
- PETTINEO, Matt. *SG Series Part 6: Step Into The Baking Lab*. Online. The Danger Zone. 10.10.2016 b. Dostupné z: <https://therealmjp.github.io/posts/sg-series-part-6-step-into-the-baking-lab/>. [cit. 2025-04-11].
- PHARR, Matt; JAKOB, Wenzel a HUMPHREYS, Greg. *Physically based rendering: from theory to implementation*. Third edition. Amsterdam: Elsevier, [2017]. ISBN 978-0-12-800645-0.
- PHONG, Bui Tuong. Illumination for computer generated pictures. Online. *Communications of the ACM*. 1975, roč. 18, č. 6, s. 311-317. ISSN 0001-0782. Dostupné z: <https://doi.org/10.1145/360825.360839>. [cit. 2024-12-21].
- Přispěvatelé Giant Bomb. *Gouraud Shading (Concept) - Giant Bomb*. Online. *Giant Bomb*. Dostupné z: <https://www.giantbomb.com/gouraud-shading/3015-4864>. [cit. 2024-12-21].
- RAHTEENKO, Arthur. *Baking artifact-free lightmaps on the GPU*. Online. Mr F. 2018. Dostupné z: <https://ndotl.wordpress.com/2018/08/29/baking-artifact-free-lightmaps/>. [cit. 2024-12-30].
- RAMAMOORTHY, Ravi a HANRAHAN, Pat. An efficient representation for irradiance environment maps. Online. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 2001, s. 497-500. ISBN 158113374X. Dostupné z: <https://doi.org/10.1145/383259.383317>. [cit. 2025-03-02].
- SCHÄFER, Henry; SÜSSMUTH, Jochen; DENK, Cornelia a STAMMINGER, Marc. Memory efficient light baking. Online. *Computers & Graphics*. 2012, roč. 36, č. 3, s. 193-200. ISSN 00978493. Dostupné z: <https://doi.org/10.1016/j.cag.2011.12.001>. [cit. 2024-12-21].
- SCRATCHPIXEL. *Introduction to Lighting – An Introduction to Lighting in 3D Rendering*. Online. Scratchapixel. Nedatováno a. Dostupné z: <https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-lighting/introduction-to-lighting.html>. [cit. 2025-01-10].
- SCRATCHPIXEL. *Introduction to Lighting – Area Lights: Mathematical Foundations*. Online. Scratchapixel. Nedatováno b. Dostupné z: <https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-lighting/introduction-to-lighting-area-lights-theory.html>. [cit. 2025-01-10].
- SEYB, Dario; SLOAN, Peter-Pike; SILVENNOINEN, Ari; IWANICKI, Michał a JAROSZ, Wojciech. The design and evolution of the UberBake light baking system. Online. *ACM Transactions on Graphics*. 2020, roč. 39, č. 4. ISSN 0730-0301. Dostupné z: <https://doi.org/10.1145/3386569.3392394>. [cit. 2024-12-21].
- SPEIGHT, Toby a , grek40. *Find nth SET bit in an int*. Online. Stack Overflow. 3.3.2017, 3.3.2017. Dostupné z: <https://stackoverflow.com/a/45484683>. [cit. 2025-04-06].
- TEO, Bernard. *Native File Dialog Extended*. Online. GitHub. Dostupné z: <https://github.com/btzy/nativefiledialog-extended>. [cit. 2025-04-10].

- THE KHRONOS GROUP. *GLTF*. Online. GitHub. Nedatováno a. Dostupné z: <https://github.com/KhronosGroup/glTF>. [cit. 2025-04-10].
- THE KHRONOS GROUP. *Sponza*. Online. GitHub. Nedatováno b. Dostupné z: <https://github.com/KhronosGroup/glTF-Sample-Assets/tree/b35907addda93c50ba655f5ac8917d73b6785302/Models/Sponza>. [cit. 2025-04-11].
- THE KHRONOS GROUP. *OpenGL - The Industry Standard for High Performance Graphics*. Online. OpenGL. Nedatováno c. Dostupné také z: <https://www.opengl.org/>.
- UNITY TECHNOLOGIES. *Lightmap data format*. Online. Unity - Manual. 2019, 26.4.2019. Dostupné z: <https://docs.unity3d.com/Manual/Lightmaps-TechnicalInformation.html>. [cit. 2025-04-10].
- UNITY TECHNOLOGIES. *Lightmapping settings in the Lighting Settings Asset reference*. Online. Unity - Manual. Dostupné z: <https://docs.unity3d.com/6000.0/Documentation/Manual/Lightmaps-reference.html>. [cit. 2025-04-10].
- VÄÄNÄNEN, Pekka. *Classic 3D videogame shadow techniques*. Online. 25.10.2024, 31.10.2024. Dostupné z: <https://30fps.net/pages/videogame-shadows/#lightmaps>. [cit. 2025-04-13].
- VEACH, Eric. *Robust Monte Carlo Methods for Light Transport Simulation*. Disertace. Stanford University, 1998.
- WÄCHTER, Carsten a BINDER, Nikolaus. A Fast and Robust Method for Avoiding Self-Intersection. Online. In: HAINES, Eric a AKENINE-MÖLLER, Tomas (ed.). *Ray Tracing Gems*. Berkeley, CA: Apress, 2019, s. 77-85. ISBN 978-1-4842-4426-5. Dostupné z: https://doi.org/10.1007/978-1-4842-4427-2_6. [cit. 2025-04-03].
- XIAOYAO WEN GONGZUO SHI. *Flat, Gouraud, Phong Shading De chabie (Comparison flat, Gouraud, Phong shading)*. Online. Xiaoyao Wen Gongzuo Shi. 1.11.2011. Dostupné z: <https://cg2010studio.com/2011/11/01/flat%E3%80%81gouraud%E3%80%81phong-shading%E7%9A%84%E5%B7%AE%E5%88%A5-comparison-flat-gouraud-phong-shading/>. [cit. 2025-02-27].
- YOUNG, Jonathan. *Xatlas*. Online. GitHub. Dostupné z: <https://github.com/jpcy/xatlas>. [cit. 2025-04-10].