Akademie múzických umění v Praze

FILMOVÁ A TELEVIZNÍ FAKULTA

Filmové, televizní a fotografické umění a nová media

Kamera

Diplomová práce

Aspekty svícení ve virtuálním prostoru

David Ticháček

Vedoucí práce: doc.Mgr. Antonín Weiser

Oponent práce:

Datum obhajoby:

Přidělovaný akademický titul: MgA.

Praha, 2022

Academy of performing arts in Prague

FILM AND TV SCHOOL

Film, Television, Photography and New Media

Cinematography

Diploma thesis

Aspects of lighting in virtual space

David Ticháček

Thesis supervisor: doc.Mgr. Antonín Weiser

Opponent:

Defense date:

Awarded title: MgA.

Prague, 2022

Prohlášení

Prohlašuji, že jsem diplomovou práci na téma

Aspekty svícení ve virtuálním prostoru

vypracoval samostatně pod odborným vedením vedoucího práce a s použitím uvedené literatury a pramenů.

Praha, dne ..... podpis diplomanta .....

Upozornění

Využití a společenské uplatnění výsledků diplomové práce, nebo jakékoliv nakládání s nimi je možné pouze na základě licenční smlouvy tj. souhlasu autora a AMU v Praze

Poděkování

Rád bych touto cestou poděkoval doc. Mgr. Antonínu Weiserovi za ochotu, vstřícnost a odborné zázemí při vedení mé diplomové práce. Dále bych rád poděkoval své rodině a přátelům za podporu a trpělivost.

Evidenční list

Uživatel stvrzuje svým podpisem, že tuto práci použil pouze ke studijním účelům a prohlašuje, že jí vždy řádně uvede mezi použitými prameny.

Jméno	**Instituce**	**Datum**	**Podpis**

Abstrakt

Ve své diplomové práci se zabývám způsoby, jakými je možné přistupovat k zasvětlování scén ve virtuálním prostoru. Snažím se předložit celkový pohled na problematiku simulace osvětlení v real-time aplikacích z pohledu tvůrce osvětlení scény, tedy člověka, který spoluvytváří estetickou kvalitu obrazu. Nejprve v práci popisuji základní principy vytváření počítačem generovaného obrazu, základní mechanismy simulace osvětlení a metody, které se využívají v real-time aplikacích pro dosažení požadovaných vizuálních výsledků. V praktické části mé diplomové práce testuji fungování nejdůležitějších principů generování obrazu a simulace osvětlení na několika příkladech virtuálních scén a porovnávám dosažené výsledky mezi sebou i s referenční fotografií. Cílem je najít optimální cesty vytváření simulovaného osvětlení ve virtuálních scénách s ohledem na výpočetní náročnost, omezení z použitých metod plynoucích a estetické kvality, kterých dané postupy dosahují.

Abstract

My diploma thesis deals with the ways in which it is possible to approach the lighting of scenes in virtual space. It attempts to present an overall view of the lighting simulation in real-time applications from the point of view of the scene lighting artist, i.e., the person who co-creates the aesthetic quality of the image. First, in the work I describe the basic principles of creating a computer-generated image, the basic mechanisms of lighting simulation, and the methods that are used in real-time applications to achieve the desired visual results. In the practical part of my diploma thesis, I analyse the functioning of the most important principles of image generation and lighting simulation on several examples of virtual scenes and compare the achieved results with each other and with a reference photo. The goal is to find optimal ways of creating simulated lighting in virtual scenes, considering the computational complexity, limitations arising from the methods used and the aesthetic quality achieved by the given procedures.

Obsah

Úvod

Dnešní film se výrazně proměňuje. Zejména postupy využívající k tvorbě či úpravám obrazu počítačem generovaný obraz velmi ovlivňují výsledný estetický dojem i způsob vyprávění příběhu. Obzvláště na dnešní hollywoodskou produkci mají silný dopad techniky, jako jsou motion capture a obecně CGI (computer-generated imagery). V poslední době se také rozmáhá takzvaná virtuální produkce, tedy způsob nahrazující zelené pozadí pomocí stěn osazených led obrazovkami, které v reálném čase na základě polohy kamery v prostoru zobrazují pozadí scény. Díky těmto technikám je možné natočit scény tak, jak by to dříve bylo nemyslitelné. Již není potřeba vytvářet precizní reálné modely či stavět celé dekorace. Vše je možné vytvořit v počítači digitálně, a to včetně simulace osvětlení scény. V případě použití tzv. virtuální produkce je dokonce možné dělat jisté úpravy přímo na place při natáčení. Například měnit polohu objektů v prostoru, měnit denní dobu, směr světla atd. Také je možné využívat světlo, které emitují led panely k částečnému nasvícené reálně části scény.

Práce při zesvětlování scény ve virtuálním prostoru má svá výrazná specifika. Tvůrci dává možnosti, o kterých by se kameramanovi při natáčení filmu ani nesnilo. Světelné zdroje, které ve virtuálním prostoru používáme mohou mít neomezenou intenzitu a velikost, mohou napodobovat chování světla reálného, ale mohou také jeho vlastnosti výrazně upravovat. Například jejich intenzita nemusí ubývat se čtvercem vzdálenosti, ale podle jiného vzorce. Můžeme ovlivňovat způsob, jakým se světlo chová po dopadu na různé materiály, nebo po jejich průchodu. Světelné zdroje ve virtuálním prostoru nepotřebují žádné stativy, můžeme s nimi libovolně pohybovat, a dokonce nemusejí být vůbec vidět. Při reálném natáčení bez použití triků je úkolem, který vyžaduje sám o sobě velkou dávku kreativity, umístit zdroje světel tak, aby vytvářely požadovaný efekt, ale zároveň nebyly pro kameru vidět (pokud se nejedná o scénická světla). Ve virtuálním prostoru zdroje nikdy vidět nejsou, pokud přímo nevytvoříme model, který by je reprezentoval.

Všechny tyto nástroje dávají tvůrci při tvorbě velkou dávku volnosti, která ale nemusí být vždy ku prospěchu. Omezení, které před sebou kameraman běžných filmů má, v něm pobízí kreativitu při hledání nových cest při řešení problémů. Z mého pohledu se často filmy využívající CGI stávají více přehledné a zobrazující vše i ve scénách, kdy to není ku prospěchu. Nejspíše částečně proto, aby se staly efektnější a bylo tak možné na tyto na první pohled přitažlivé scény lákat diváky, nebo prostě proto, že v dnešní době lze perfekcionismu dosáhnout. Snahou již není zakrýt nedokonalosti, ale ukázat dokonalost. Aby se toho tvůrce vyvaroval, musí si, podle mého názoru, překážky a omezení klást sám sobě.

Prací kameramana u filmu je ovlivňovat podvědomí diváka a manipulovat s jeho emocemi tak, aby techniky k tomu použité, divák primárně nevnímal. Jedná se o velmi výrazné i drobné výrazové prostředky. Od způsobu nasvícení scény, pohybu kamery, kompozice až po volbu objektivu. Stejné prostředky může tvůrce používat při vytváření vizuální stránky videohry, nebo jakéhokoli jiného audiovizuálního díla, které využívá počítačem generovaný obraz. Tyto prostředky mohou mít podobný účinek na hráče, jako na diváka filmu. Zvýšit celkový dojem a emoční působnost.

Ať už se tvůrce zabývá vytvářením počítačem generovaného obrazu pro film či videoherní průmysl je pro něj nutností chápat principy, jakými se obraz generuje a jaké úskalí tyto technologie skýtají. Jen v momentě, kdy člověk ovládne techniku, kterou používá, tak ho

přestane svazovat. Začne pracovat svobodně a dokonalost již nebude cílem, ale pouze alternativou. Chyba nebude nepřítelem, ale chtěnou součástí celku. Tvůrce se nebude snažit pouze ohromit, ale vzbudit daleko hlubší emoce.

V následujících kapitolách mé diplomové práce se snažím podat ucelený pohled na problematiku simulace osvětlení ve virtuálním prostoru pomocí real-time aplikací. Objasnit základní principy, které real-time aplikace využívají a najít vhodné technologie pro dané použití porovnáváním reálných scén se scénami vytvořenými v počítači pomocí různých metod a technologií.

I. Teoretická část

1 | Simulace osvětlení ve virtuálním prostoru

Pro pochopení rozdílných přístupů při zasvětlování scény reálné a scény vytvořené v počítači je nutné pochopit základní principy, na kterých zobrazování počítačové grafiky funguje. Rozdíly mezi zasvětlováním prostoru obou realit jsou nejen v rozšířených možnostech, které virtuální prostor tvůrci dává, ale také v různých omezeních, které plynou jak z principů fungování zobrazení 3D grafiky, tak z limitů výpočetních výkonů dnešních počítačů. Následující kapitoly popisují základní principy fungování zobrazování prostorových objektů a simulaci jejich nasvícení. Také vysvětluje pojmy, se kterými se dále v textu pracuje.

1.1 Zobrazování prostorových objektů

Zobrazování prostorových objektů pomocí počítače se obvykle nazývá renderování. Ve 3D grafice je to proces, při kterém program převádí vstupní data do podoby rastrového obrazu.[1] Mezi tyto data patří zejména 3D modely, světelné zdroje, shaderové algoritmy, textury, pozice, atributy virtuální kamery atd.

Velmi zjednodušeně lze proces renderování popsat otázkou „Jakou barvu má tento pixel?“. Tuto otázku renderovací program musí položit mnoha milionkrát pro každý snímek. K tomu, aby na ní mohl odpovědět potřebuje znát tři základní údaje: kolik světla dopadá na daný bod v prostoru, jak objekt v daném bodě odráží či propouští světlo, případně zda nějaké světlo emituje a kde se nachází virtuální kamera. K tomuto výpočtu používá renderovací program takzvanou renderovací rovnici.

$$L_o(\omega_o)=L_e(\omega_o)+\int_\Omega f(\omega_i,\omega_o)L_i(\omega_i)(\omega_i\cdot n)d\omega_i \quad [2]$$

Proces renderování, v závislosti na použité metodě, komplexnosti scény a hardwaru, může trvat pro každý snímek od zlomku sekundy až po několik dní. Aplikace, které jsou podmíněné interakcí s uživatelem však musí tento proces zvládnout v řádu setin sekundy tak, aby výsledný sled obrázků vytvořil iluzi plynulého pohybu. Obvykle se jako spodní limit pro plynulý obraz považuje přibližně 25 fps (frames per second) a jako horní limit 120 fps. Horní limit je počet obrázků za sekundu, nad který už pozorovatel nerozezná žádný rozdíl v plynulosti pohybu.

[1] KLAWONN, F. M. Introduction to Computer Graphics Edtion ed. London: Springer, 2008. ISBN: 978-1-84628-847-0.

[2] HAINES, E. Ray Tracing Essentials, Part 1: Basics of Ray Tracing. NVIDIA, 2019

Tento údaj je ovšem velmi přibližný a závisí na každém pozorovateli, zejména na jeho věku. Obecně platí, že více než 50% populace dokáže rozeznat více než 45 fps.[3]

Kvůli požadavku počítačových her a jiných real-time aplikací generovat desítky obrázků za sekundu je nutné značně optimalizovat zdrojová data a využívat techniky, které vedou k co nejnižšímu zatížení hardwaru při zachování dostatečné kvality výsledného obrazu. Mezi tyto techniky patří zejména systém LOD (level of detail), object instancig, texture maping, Light Baking atd., kterým se věnují pozdější kapitoly.

1.2 Způsoby iluminace

Rozhodujícím faktorem pro výsledný dojem z obrazu a pro hardware výpočetně nejnáročnějším úkolem renderovacího programu je výpočet matematické reprezentace distribuce světla ve scéně. V následujících kapitolách se budu věnovat způsobům simulace iluminace a popisovat, v jakém jsou vztahu k fyzikálním vlastnostem reálného světla. Také přiblížím nejpoužívanější principy renderování jako je scanline rendering a ray-tracing.

Způsoby iluminace virtuálního prostoru můžou být rozděleny na dva základní způsoby. Prvním je local illumination, který počítá světlo, které dopadá na objekt přímo ze světelného zdroje. Dále počítá, jak povrchy objektů reagují na dopadající světlo. To zahrnuje odrazy od povrchu, průchod světla povrchem a tzv. subsurface scattering, který popisuje, jak se světlo odráží pod povrchem objektu, než ho opustí. Druhým základním způsobem iluminace je tzv. global illumination, který počítá iluminaci scény jako celku. Zejména jde o světlo, které dopadá na povrchy nepřímo, z odrazů a rozptylů od ostatních předmětů. Je zřejmé, že dobrý model global illumination nemůže fungovat správně bez dobře fungujícího modelu local illumination. Oba modely jsou propojené a navzájem se ovlivňují a není zde zcela jasná hranice mezi nimi.[4]

[3] HUMES, L. E., BUSEY, T. A., CRAIG, J. C., KEWLEY-PORT, D.
Atten Percept Psychophys. Bloomington, Indiana: Indiana University, 2009.
doi: 10.3758/APP.71.4.860.

[4] KURACHI, N. The Magic of Computer Graphics. Edtion ed. Tokyo, Japan: Ohmsha, Ltd.,, 2007. ISBN: 978-1-56881-577-0.

1.2.1 Global illumination

Scény, které využívají global illumination působí obvykle daleko realističtěji než ty, které jsou osvětleny výhradně za použití local illumination. global illumination je ovšem velmi náročný na výpočetní výkon a generování snímků může trvat velmi dlouho. Extrémní případ, kdy je global illumination zásadní pro výsledný vzhled obrazu je scéna, kde je většina objektů nasvícena pouze světlem odraženým od okolních objektů. Příklad takové scény můžeme vidět na obr. 1.2-2 a obr. 1.2-1. Světelným zdrojem je slunce, které úzkou mezerou mezi skalami proniká až na dno průrvy. Skalnaté stěny jsou tak osvětleny téměř výhradně od světla odraženého. V menší míře i světlem rozptýleným v atmosféře. Oba tyto zdroje, tedy světlo rozptýlené v atmosféře i odražené od jiných objektů nazýváme globálním nebo sekundárním osvětlením. Pokud renderovací program nepočítal sekundární osvětlení, byly by stěny průrvy zcela černé nehledě na nastavení „expozice“ virtuální kamery.

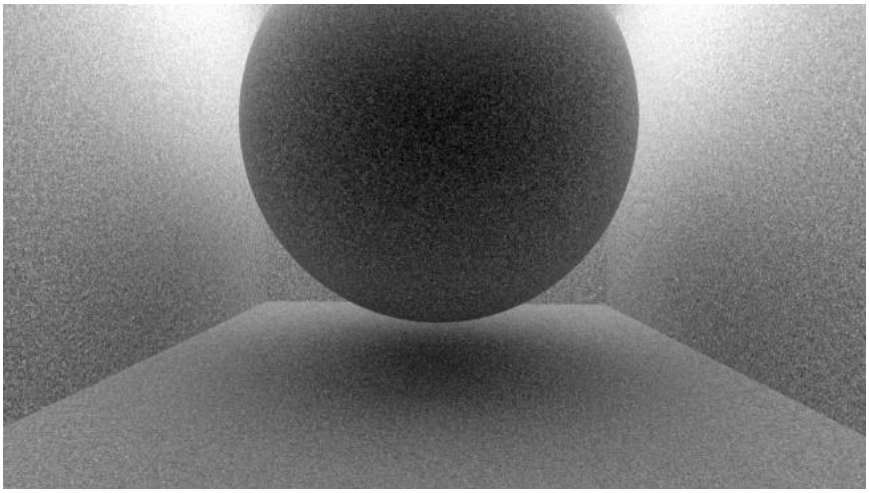
obr. 1.2-2 pouze local illumination
vlastní tvorba (Houdini 18.5 + Mantra)

obr. 1.2-1 local illumination + global illumination
vlastní tvorba (Houdini 18.5 + Mantra)

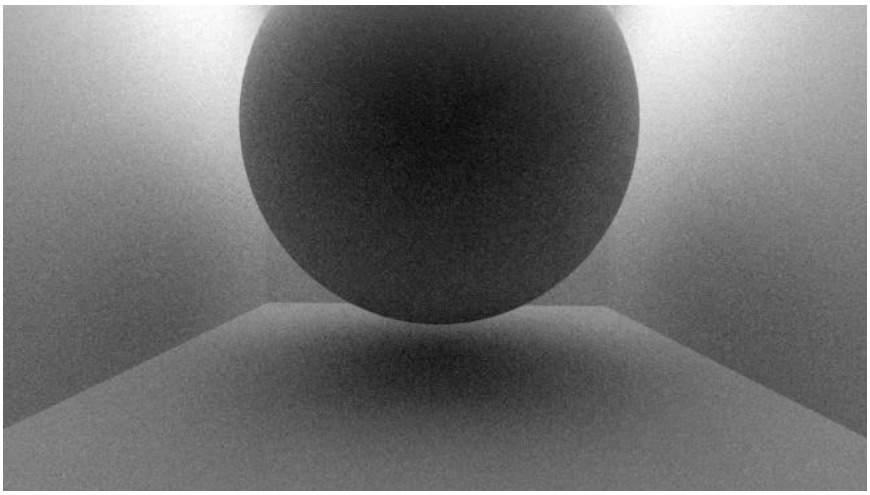
Existuje mnoho algoritmů pro výpočet global illumination. Většina renderovacích programů obsahuje několik těchto algoritmů, ze kterých si uživatel může vybrat ten, který se nejvíce hodí pro danou scénu a požadovaný výsledek. Často se kombinuje více algoritmů, pro dosažení co možná nejlepšího výsledku v co nejkratším čase. Každý z algoritmů má své přednosti a nedostatky. Některé algoritmy, jako například Brute Force jsou výpočetně mnohonásobně náročnější než jiné. Dosahují však stálejších výsledků a jsou proto jsou vhodné pro renderování animací, kde díky tomu nedochází k tzv. flickerování, tedy nekonzistentnosti mezi sousedními snímky v animaci. Při výběru algoritmu je také nutné brát v úvahu komplexnost scény a

případné speciální efekty. Například algoritmus Photon mapping většinou nebere v úvahu světlo emitující materiály. Obrázky obr. 1.2-3 a obr. 1.2-4 demonstrují různé výsledky odlišných algoritmů při zasvícení scény pouze odraženým světlem při stejném čase renderování obou snímků.

Pro dosažení realistického vzhledu většiny scén je global illumination nezbytný. V real-time aplikacích není možné kvůli výpočetní a časové náročnosti global illumination používat stejným způsobem, jako se to děje v non-real-time renderech. Proto bylo vyvinuto mnoho způsobů, jak global illumination simulovat bez nutnosti přímého výpočtu velkého množství odražených paprsků v reálném čase. Konkrétním způsobům simulace dynamického globálního osvětlení se věnuji v kapitole 2.2



obr. 1.2-3 Micropolygon rendering – renderovací čas 1minuta 20sekund (vlastní tvorba) (Houdini 18.5 + Mantra)



obr. 1.2-4 Ray Tracing – renderovací čas 1minuta 20sekund (vlastní tvorba) (Houdini 18.5 + Mantra)

1.2.2 Local illumination

Local illumination je koncept algoritmů, které počítají pouze světlo, jehož paprsky dopadají přímo ze světelného zdroje na 3D model a od něho se odrážejí na obrazovou rovinu virtuální kamery. Tyto algoritmy musí vypočítat, jakým způsobem je světlo odraženo, pohlceno a propuštěno objektem. Nepočítá však světlo odražené od okolních objektů. To, jak světlo reaguje po dopadu na objekt je definováno tzv. pixel shadery. Neboli kódy a algoritmy, které ovlivňují barvu každého pixelu v závislosti na jeho virtuálním materiálu a světlu na něj dopadajícím.[5] Materiály 3D objektu pak mohou obsahovat texturové mapy a matematické operace ovlivňující vlastnosti materiálu.

1.3 Metody renderování

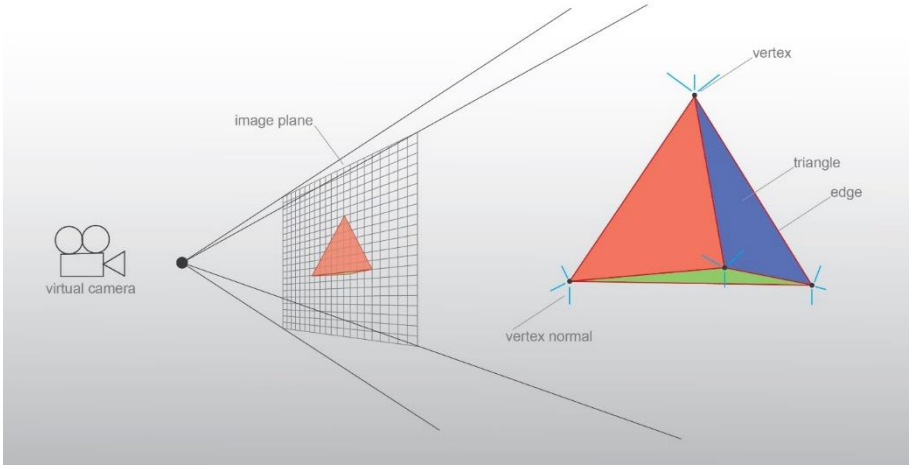
Za dva hlavní tipy algoritmů, pomocí kterých se v počítačové grafice generuje 2D obraz z 3D objektů se považuje rasterizace a ray-tracing. Ačkoli obě metody vznikly přibližně ve stejnou dobu, rasterizace se rychle stala dominantní zejména ve interaktivních aplikacích, díky své relativní výpočetní nenáročnosti a možnosti ji snadno přesunout na hardware. Nevýhodou rasterizace je však složité implementování globálních efektů jako jsou například odrazy.[6] Metoda ray-tracing byla dlouho považována jako nekonkurenceschopná v interaktivních 3D aplikacích kvůli její výpočetní náročnosti. Se výrazně se zvyšujícím výpočetním výkonem grafických procesorů a díky úspěšné snaze o její namapování na hardware se stává stále častěji používaná i v interaktivních aplikacích.

1.3.1 Rasterizace

Objekt ve virtuální scéně se skládá z tzv. vertexů (body v prostoru), které reprezentují vrcholy trojúhelníků. Z těchto trojúhelníků je tvořen plášť daného objektu. Pro zobrazení 3D objektů na rovinnou plochu musíme také znát místo v prostoru, ze kterého scénu pozorujeme, úhel pod kterým ji pozorujeme a zorné pole. Soubor těchto informací nazýváme virtuální kamerou. Ve virtuálním prostoru pak vzniká obrazové pole, které si můžeme představit jako obrazovku či snímač kamery. Vertexy, z nich tvořené trojúhelníky a virtuální kamera, respektive obrazové pole, jsou tři součásti nutné pro proces jednoduché rasterizace. Viz. obr. 1.3-1.

[5] MIKE BAILEY, S. C. *Graphics Shaders Theory and Practice*. Edtion ed. New York: CRC Press, 2012. ISBN 978-1-4398-6775-4.

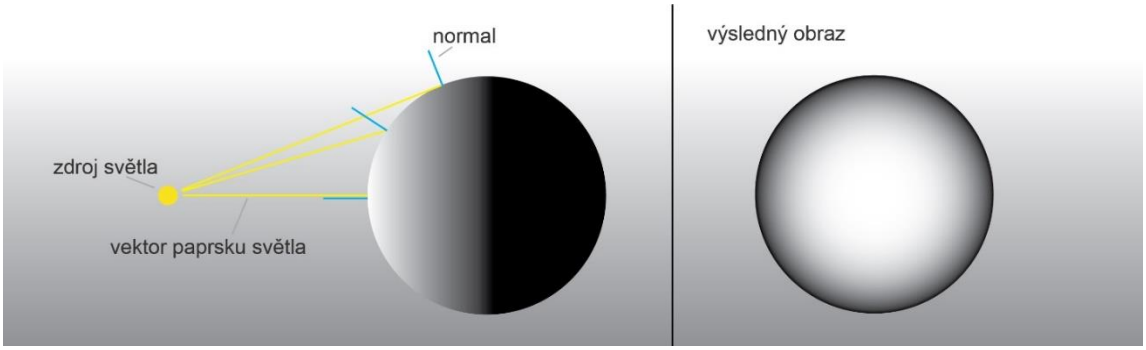
[6] SLUSALLEK P., GEORGIEV, I., ENGELHARDT, T., PHILLIPS, M., DAVIDOVIC, T., DACHSBACHER, C., "3D Rasterization–Unifying Rasterization and Ray Casting." (Universität des Saarlandes) 2009.



obr. 1.3-1 Znázornění principu rasterizace. (lastní tvorba)

Princip rasterizace si můžeme představit tak, jako kdybychom jednotlivé trojúhelníky objektů házeli na obrazové pole. Na obrazovém poli algoritmus určí, zda střed každého pixelu leží vně trojúhelníku, či nikoliv. Tak vznikne 2D reprezentace v prostoru ležícího trojúhelníku. Do mezipaměti se uloží vzdálenost každého vertexu od virtuální kamery (tzv. Z-buffer). Díky této vzdálenosti pak program zobrazí pouze trojúhelníky, či jejich části, které nejsou zakryté jiným trojúhelníkem. Celý tento proces je přenesen na hardware grafického čipu a není tedy možné ho softwarově přeprogramovat.[7]

To, jakým způsobem objekty reagují na světlo má na starost kód, kterému se obecně říká shader a který je již zcela programovatelný. Aby algoritmus poznal, zda na daný trojúhelník dopadá světlo či nikoli a pod jakým úhlem, potřebujeme ještě informaci, jakým směrem je každý trojúhelník orientován. K tomu slouží mimo jiné i tzv. vertex normal. Pro každý vertex a zároveň pro každý vrchol trojúhelníku existuje vektor, pomocí kterých program zjistí orientaci trojúhelníku. Podle úhlu, který trojúhelník svírá s vektorem paprsků vycházejících ze světelného zdroje se určí, zda je trojúhelník ve stínu, či nikoliv. Viz obr. 1.3-2.



obr. 1.3-2 Znázornění základního principu výpočtu nasvícení objektu při rasterizaci(vlastní tvorba)

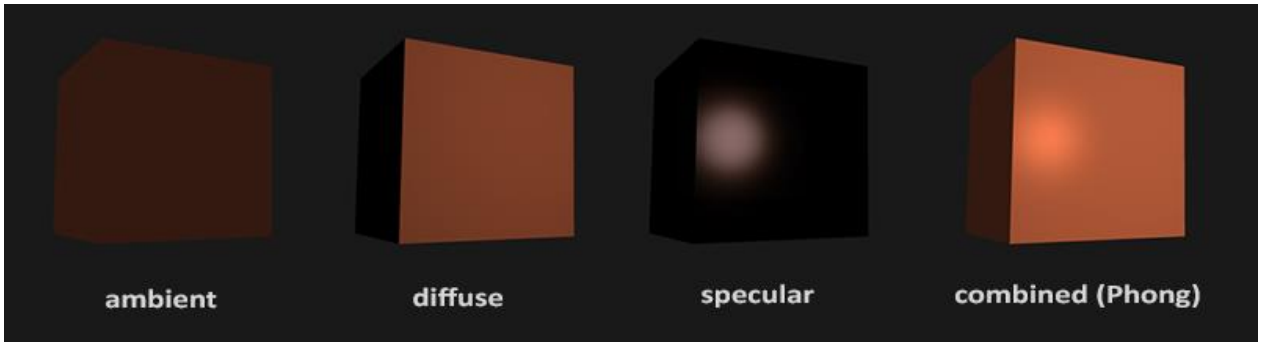
[7] CAULFIELD, B. Nvidia Blogs. In., 2018.

1.3.2 Phongův osvětlovací model

Důvod, proč objekty v reálném světě vypadají tak, jak vypadají je často kvůli velmi složitým interakcím mezi světlem a mikroskopickým strukturám materiálu. Bylo by příliš komplikované tyto interakce simulovat v počítači, a proto se používá matematický model, který tyto interakce přibližně napodobuje.

Phongův osvětlovací model je model založen na pozorování reality, nevychází z fyzikálních vlastností světla. Výsledky, kterých dosahuje jsou však tak dobré, že se v počítačové grafice používá již několik desetiletí. Zejména real-time aplikace Phongův model a jemu příbuzné modely používají kvůli jeho výpočetní nenáročnosti.

Phongův model je zaležen na pozorování, že na lesklých površích se zobrazují malé intenzivní body světla, zatímco na matných površích se vytváří větší světlé plochy menší světelné intenzity, která klesá pozvolněji. Proto je Phongův osvětlovací model rozdělen do tří hlavních světelných komponent: Ambient, Diffuse a Specular. Pro každý objekt je každá tato komponenta vypočítána zvlášť a následně jsou spojeny do výsledné podoby nasvíceného objektu.[8]
[9]



obr. 1.3-3 Krychle zobrazené vždy jednou ze složek Phongova osvětlovacího modelu. Poslední krychle zprava zobrazená kombinací všech tří součástí Phongova modelu. (Vries, Basic Lighting nedatováno)

Ambient lighting, neboli ambientní světlo je složka Phongova modelu, která simuluje světlo rozptýlené od okolních objektů. Model pracuje s předpokladem, že obvykle vidíme i ty části objektů, které nejsou nasvícené přímím světlem. Intenzita ambientního světla je ve všech bodech objektu stejná. Můžeme však měnit intenzitu ambientního osvětlení a odrazivý koeficient ambientního světla každého objektu.

Diffuse light, neboli difusní světlo je složka Phongova modelu, která je zodpovědná za největší vizuální dopad na zobrazované modely. Objektům dává dojem plasticity. Světlo, dopadá ze směru světelného zdroje a od povrchu se odráží rovnoměrně všemi směry. Tato složka Phongova modelu simuluje světlo, které dopadá na matný předmět. Intenzita odraženého světla není závislá na úhlu virtuální kamery, protože světlo se od povrchu odráží do všech směrů rovnoměrně. Je však závislá na úhlu „dopadu paprsku“ na objekt. Čím blíže je úhel dopadu k normále povrchu, tím větší intenzitu difuzní světlo má.

[8] VRIES, J. D. Basic Lighting. In *Learn OpenGL.*

[9] VRIES, J. D. *Learn OpenGL: Learn modern OpenGL graphics programming in a step-by-step fashion.* Edtion ed.: Kendall & Welling, 2020. ISBN 9090332561.

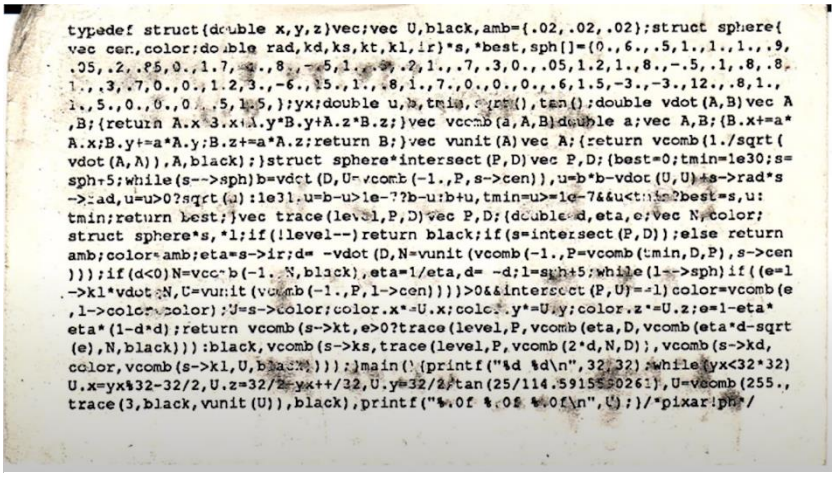
Specular light, neboli lesklé světlo, je složka Phongova modelu, která dává intenzitu světlu, které se odráží zejména v jednom směru podle zákona odrazu světelného paprsku. Tato složka je, co se barvy týče, více závislá na barvě světelného zdroje než na barvě objektu.

Celkové nasvícení objektu se získá pouhým sečtením jednotlivých složek Phongova modelu.[10]

1.3.3 Ray-tracing

Základní princip fungování ray-tracingu je velmi prostý. Kód jednoduchého algoritmu ray-tracingu si dokonce napsal vývojář Pixaru Paul Heckbert na zadní stranu vizitky. Jeho dnešní aplikace jsou daleko komplexnější a často se využívá v kombinaci s jinými metodami renderování.

[11]



obr. 1.3-4 Vizitka Paula Heckberta (cca 1980) (Haines 2019)

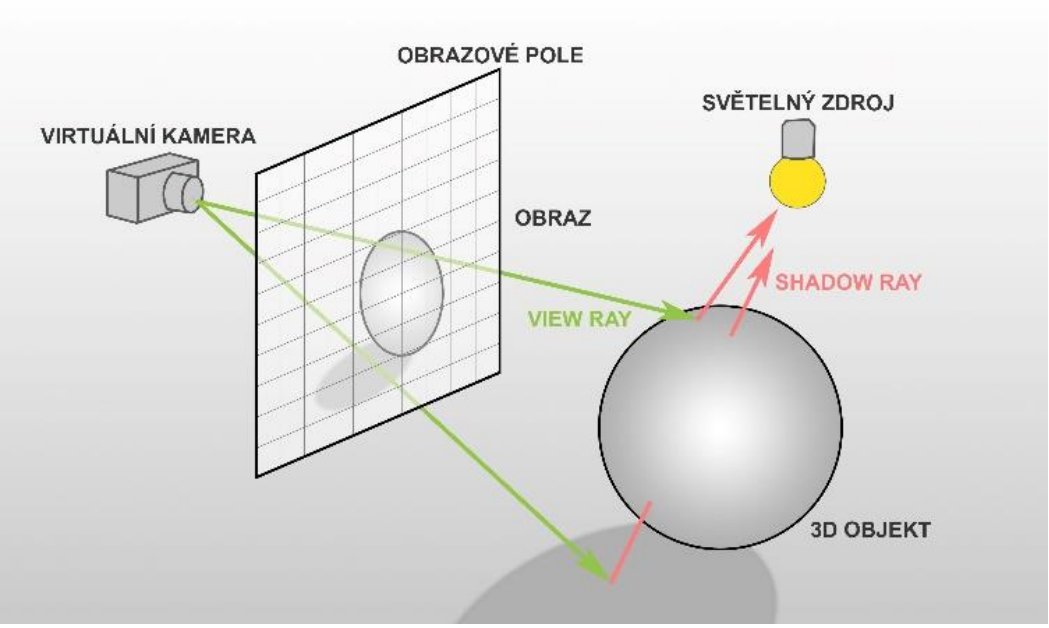
Ray-tracing je založený na co nejpřesnější simulaci fyzikálních vlastností světla. Reprodukuje chování paprsků světla, jejich odrazy od předmětů, průniky materiály, refrakce atd. V realitě putuje paprsek světla ze světelného zdroje a přes odrazy se dostává na sítnici pozorovatele případně na snímač kamery. Při ray-tracingu se tento směr obrací. Paprsky, které simuluje, vycházejí z virtuální kamery a program počítá jejich průchod virtuálním prostředím až ke světelnému zdroji. Tento základní princip je zobrazen na obr. 1.3-5.

Z virtuální kamery vysíláme paprsek (tzv. view ray), přes každý pixel obrazového pole. Pokud view ray dopadne na předmět ve virtuální scéně scéně, vysíláme druhý paprsek (tzv. shadow ray) z místa dopadu na objekt směrem ke zdroji světla. Pokud shadow ray až ke světelnému zdroji doputuje, víme, že dané místo je osvětlené. Pokud shadow ray při cestě ke světelnému zdroji narazí do jiného předmětu, víme, že je dané místo ve stínu. Když tento proces opakujeme pro každý pixel, obrazového pole, získáme výsledný 2D obraz. Vysílání těchto paprsků se říká také ray casting. Jedná se o univerzální nástroj, který se dá využívat mnoha

[10] TUJULA, A. Lighting and normal mapping in computer Graphics. Turku University of Applied Sciences, 2016.

[11] HAINES, E. Ray Tracing Essentials, Part 1: Basics of Ray Tracing, NVIDIA, 2019.

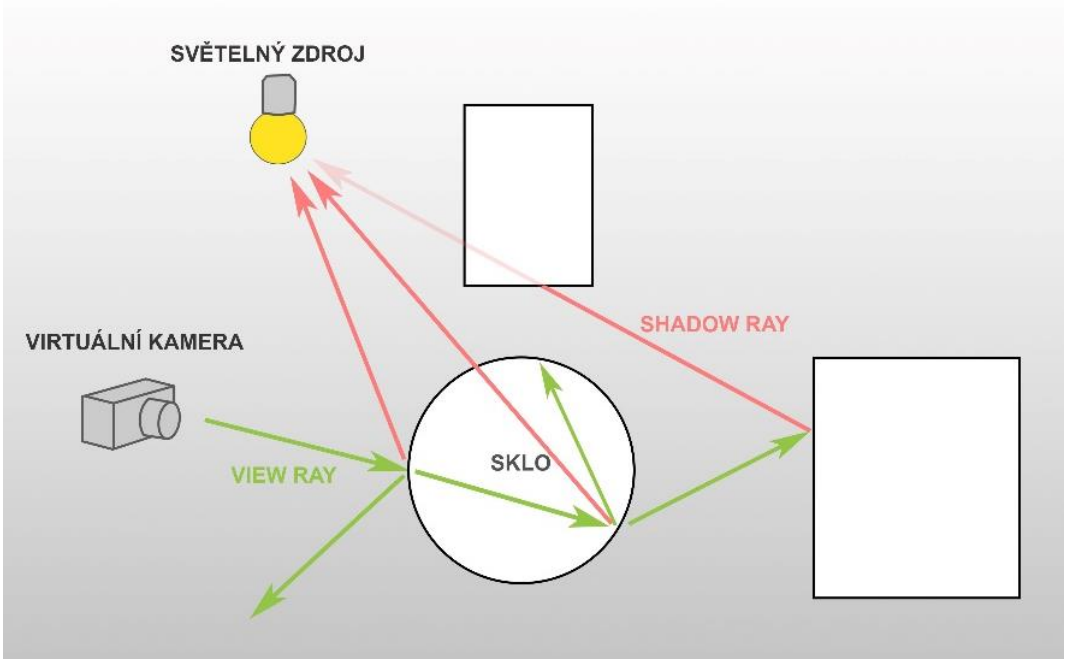
způsoby a rozšiřovat ho do komplexnějších podob, které jsou schopny vytvářet věrné zobrazení reality.
[12]



obr. 1.3-5 Znázornění principu metody renderování ray-tracing (vlastní tvorba)

Princip ray-tracingu vyniká nad rasterizací ve vizualizaci lesklých a průhledných předmětů například vody, skla nebo kovů. Reálné světlo se chová tak, že když paprsek světla dopadne například na skleněný povrch, určitá jeho část se odrazí, část se pohltí a část projde pod určitým úhlem. ray-tracing toto chování světla dokáže simulovat velmi přesně. Z místa dopadu prvního view ray paprsku na virtuální předmět, který má definovaný materiál reprezentující sklo, vyšle program nejen shadow ray, ale i další tzv. sekundární view ray paprsky. Některé skrz objekt a některé jiným směrem v závislosti na vlastnostech daného materiálu a úhlu dopadu. Pokud tyto sekundární paprsky narazí na jiný předmět, tak z místa jejich dopadu vysílá program shadow ray směrem ke zdroji světla. Pokud sekundární paprsek na žádný předmět nenarazí, program ho může ignorovat. Viz obr. 1.3-6.

[12] GLASSNER, A. S. *An Introduction to Ray Tracing*. Edtion ed. San Diego: Academic Press INC., 1989. 327 p. ISBN 9780122861604.



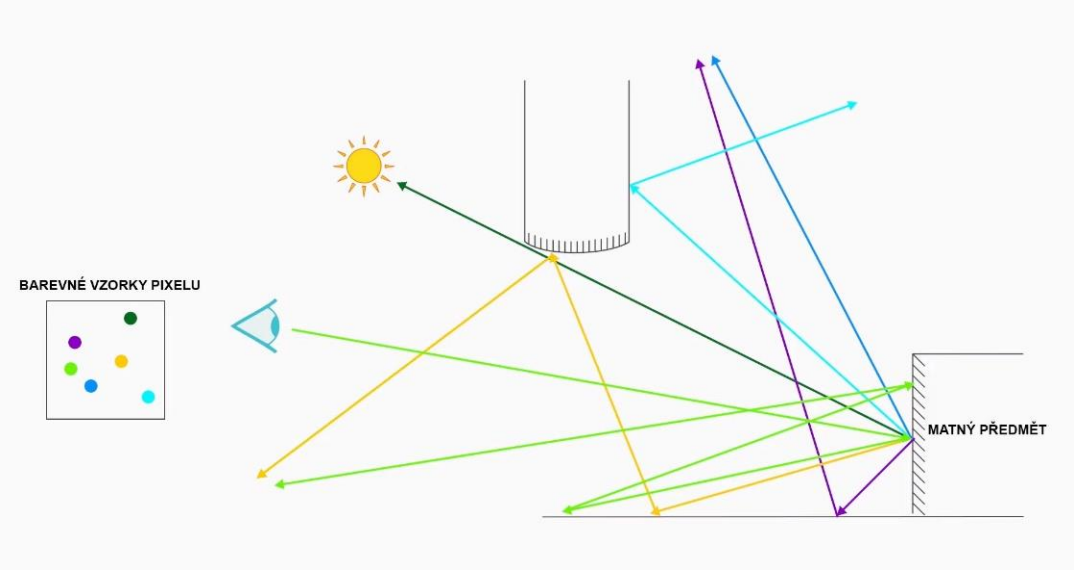
obr. 1.3-6 Znázornění principu metody renderování ray-tracing (vlastní tvorba)

Čím více sekundárních view ray program vyšle, jinak řečeno, čím více odrazů světla programu nařídíme počítat, tím bude výsledný obraz věrnější. Zároveň se však výrazně zvyšuje výpočetní náročnost. Touto cestou dokáže ray-tracing velmi věrně simulovat chování světla při průchodu různými materiály.

Jedním z podmnožiny algoritmů ray-tracingu je tzv. path-tracing. Tento způsob představuje výrazný skok v náročnosti výpočtu, ale i věrnosti zobrazení virtuální scény v některým aspektech. Path-tracing vyniká nejen při výpočtech odlesků, refrakcí a jevů, které jsou svým chováním předvídatelné, ale i světla dopadajícího a odrážejícího se od povrchů, které světelné paprsky odrážejí všemi směry náhodně. Tedy od zcela matných, jako je například beton nebo některé druhy textilií, po částečně lesklé a polo matné, jako jsou neleštěné kovy, plasty atd.

Podstata fungování metody path-tracing (obr. 1.3-7) spočívá ve vysílání velkého množství paprsků z virtuální kamery přes každý pixel obrazového pole. Podle parametrů materiálu, na který paprsky dopadají se každý z nich odrazí pod jiným úhlem. Program sleduje jeho cestu prostorem, dokud se paprsek neodrazí do světelného zdroje (nebo blízko něj) anebo dokud nedosáhne maximálního nastaveného počtu odrazů. Informace z jednotlivých vyslaných paprsků se zprůměrují do výsledné barvy pixelu. Díky náhodnému odrazu od povrchu a mnoha vyslaným paprskům získáme přesnější informaci o nepřímo dopadajícím světle a odrazech od ostatních předmětů. V praxi se při offline renderování vysílá pro každý pixel až tisíce paprsků. Taková operace výpočetně extrémně náročná, ale přesto tuto metodu používá většina profesionálních renderovacích softwarů. Věrnost zobrazení je výrazně vyšší oproti jiným renderovacím metodám zejména u simulace nepřímého osvětlení neboli global illumination.[13]

[13] HEARN, P. What is Path Tracing and Ray Tracing? And Why do They Improve Graphics? In., 2019, vol. 2021.



obr. 1.3-7 Znázornění principu metody renderování path-tracing (vlastní tvorba)

1.4 Vizuální porovnání rasterizace a ray-tracingu

Při porovnání základních podob rasterizace a ray-tracingu je rozdíl viditelný na první pohled. ray-tracing dosahuje daleko reálněji působící scény hlavně díky fyzikálně přesnějším odleskům, stínům a odrazům světla od všech povrchů viz obr. obr. 1.4-1.
[14]

obr. 1.4-1 Porovnání scén renderovaných pomocí rasterizace a ray-tracingu (Sukjun Park 2021)

Po dlouhou dobu téměř výhradní využívání rasterizace v real-time aplikacích vedlo k výraznému posunu při snaze napodobit fyzikální vlastnosti světla bez použití ray-tracingu.

[14] PARK, S. Shader-Based Ray Tracing Engine. In Applied Sciences. 2021, vol. 7.

Moderní způsoby využití rasterizace díky tomu dokáží generovat obraz podobný tomu, který generuje ray-tracing. Protože renderovacícm metodám, které využívají rasterizaci, chybí informace o paprsku a jeho „cestě“ scénou, musí vždy chování světla napodobit jiným, tedy fyzikálně nepřesným způsobem. Pro každý jev, jako jsou odrazy, refrakce, stíny, ambient occlusin (zastíněné okolím), atd., musí vývojáři použít algoritmus, který daný jev napodobí.

Následující obrázky ukazují rozdílné výsledky dosažené při použití moderních renderovacích metod využívajících rasterizaci a ray-tracing, konkrétně je použit software Blender 2.79 a renderovací systémy Evee (rasterizace) a Cycles (raytracing).

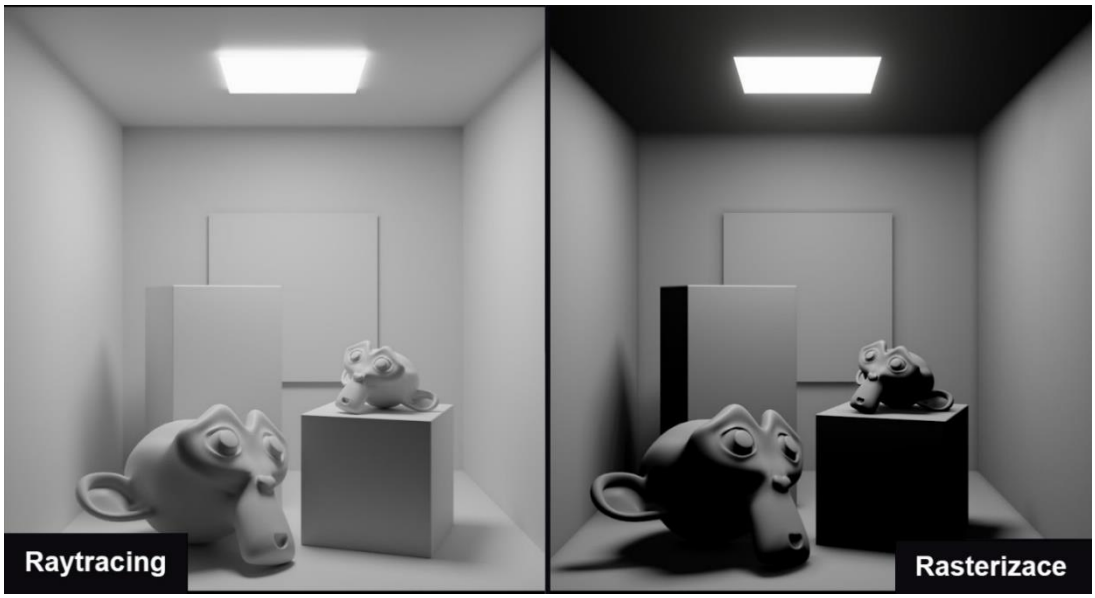
Na obrázku obr. 1.4-2 vidíme, jak ray-tracing velmi dobře zvládá, v porovnání s rasterizací, simulaci průchodu světla skleněným objektem. rasterizace sice dokáže napodobit refrakci jednoduchých objektů, jako je skleněná koule, ale s komplexnějším objektem má již problémy. [15]

obr. 1.4-2 Porovnání scén renderovaných pomocí rasterizace a ray-tracingu (Lampel 2019)

Na obrázku obr. 1.4-3 Porovnání scén renderovaných pomocí rasterizace a ray-tracingu vidíme rozdíl ve výsledku zobrazení stínů a sekundárního osvětlení při použití dvou zmíněných metod. V případě rasterizace jsou zastíněná místa příliš tmavá a bez kresby. rasterizace simuluje stíny tak, že sleduje, co vše je „vidět“ z pohledu světelného zdroje. To, co vidět není, je ve stínu. rasterizace sice může dosáhnout rozostřených okrajů stínů, jak vidíme na obrázku, ale nemůže pracovat se světelným paprskem, a tedy s odrazy světla tak, jak to dělá ray-tracing. Proto je těžké dosáhnout přirozeně působícího obrázku.

[15] LAMPEL, J. Cycles vs. Eevee - 15 Limitations of Real Time Rendering in Blender 2.8. 2019.

[16]



obr. 1.4-3 Porovnání scén renderovaných pomocí rasterizace a ray-tracingu (Lampel 2019)

Další a výhodou ray-tracingu je přesnost, se kterou dokáže simulovat odlesky. Při rasterizaci je možné různými technikami tento jev napodobit. Například pokud je lesklý předmět plochý, nebo se přiměřeně blíží plochému, je možné odraz rastrovat v separátním průchodu převrácením odraženého objektu a namapováním tohoto odrazu na danou část obrazu. Pomocí shaderu je pak možné odraz deformovat pro získání věrnějšího zobrazení. Tento postup se používá například u klidných vodních hladin. Pokud jsou vlny však příliš velké, efekt přestává být věrný.[17]

[18]

obr. 1.4-4 Porovnání scén renderovaných pomocí rasterizace a ray-tracingu (icyou520 2018)

[16] ICYOU520. Eevee vs Cycles Comparisons. 2018.
[17] REED, N. Mirror Reflections 2017.
[18] ICYOU520. Eevee vs Cycles Comparisons.2018.

1.5 Metody kombinující Raytracing a Rasterizaci

Pro dnešní běžné osobní počítače je použití ray-tracingu pro renderování celé scény v reálném čase stále příliš výpočetně náročné. Proto byly vyvinuty systémy, které kombinují rasterizaci a ray-tracing tak, aby byl ray-tracing použit pouze na některé elementy obrazu, kde přináší největší vizuální benefit oproti rasterizaci. V takovém případě ray-tracing obvykle vykresluje odrazy od lesklých povrchů, počítá globální osvětlení, zobrazuje stíny a tzv. ambient occlusion (zastínění okolím). Primární zobrazení scény je řešené stejně jako dříve přes rasterizaci.[19]

Na obrázkách obr. 1.5-1, obr. 1.5-2 a obr. 1.5-3 vidíme, že při použití ray-tracingu pouze na některé prvky scény jsou rozdíly spíše jen drobnými změnami než zcela odlišným způsobem zobrazení. Můžeme však předpokládat, že s postupně se zvyšujícím se výkonem grafických karet a lepší hardwarovou akcelerací bude ray-tracing využíván při větší části zobrazení scény, a tak bude mít i větší dopad na její celkový vzhled. Nejen, že vyobrazení scény bude více odpovídat realitě, jak ji známe, ale pro vývojáře aplikací bude daleko snadnější realismu dosáhnout. Nebude nutné napodobovat každý efekt, který světlo ve scéně vytváří mnoha rozdílnými technikami, ale bude možné simulovat fyzikální vlastnosti světla tak, jak se chová v realitě.

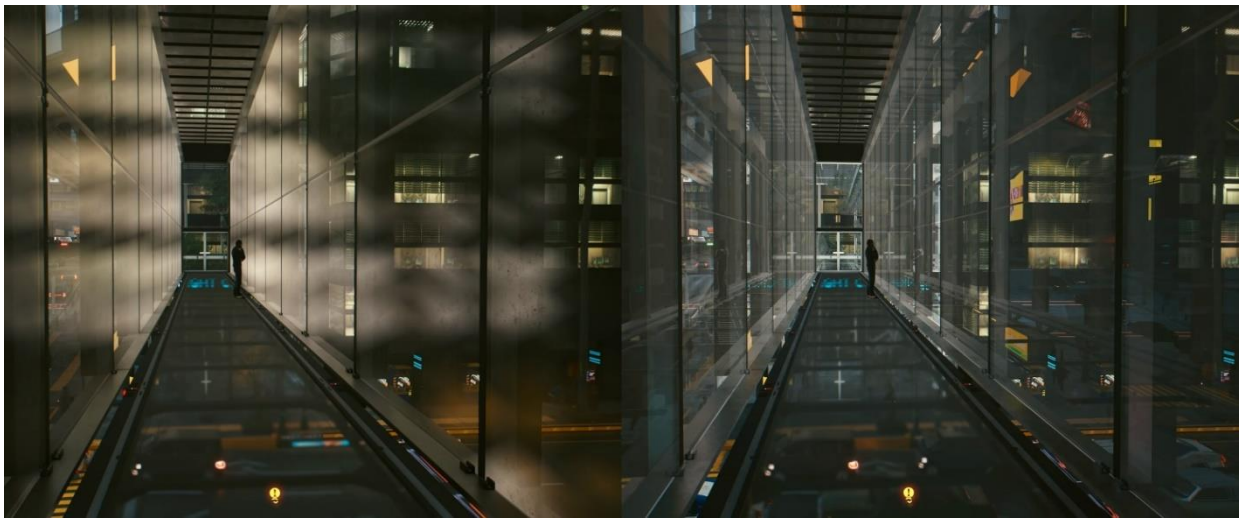
20

obr. 1.5-1 Hra Metro Exodus zobrazovaná pomocí rasterizace (vlevo) a pomocí kombinace rasterizace a ray-tracingu (vpravo) (Evga 2022)

[19] CABELEIRA, J. Combining Rasterization and Ray Tracing Techniques to Approximate Global Illumination in Real-Time. Técnico Lisboa, 2010.

[20] EVGA. Flee The Shattered Ruins Of Moscow. 2022.

[21]



obr. 1.5-3 Hra Cyberpunk zobrazovaná pomocí rasterizace (vlevo) a pomocí kombinace rasterizace a ray-tracingu (vpravo) (Cycu1 2020)

[22]

obr. 1.5-2 Hra Shadow of the Tomb Raider zobrazovaná pomocí rasterizace (vlevo) a pomocí kombinace rasterizace a ray-tracingu (vpravo) (Mundra 2021)

[21] MUNDRA, A. What Is Ray Tracing? 2021.

[22] CYCU1. Cyberpunk 2077 Ray Tracing ON vs OFF RTX 3080 4K Graphics Comparison. 2020.

2 | Aproximace sekundárních světelných zdrojů

Protože real-time aplikace využívající rasterizaci neumožňují simulovat fyzikální vlastnosti světla tak, jak to dělá ray-tracing, je nutné k dosažení realistického zasvícení scény využívat různých metod, které chování světla napodobují. Simulace přímého světla je výpočetně poměrně nenáročná, ale pro naprostou většinu scén je použití pouze přímého osvětlení nedostatečné. Sekundární osvětlení, tedy osvětlení přicházející od přímo i nepřímo nasvícených předmětů, světlo rozptýlené v atmosféře či světlo vycházející z jiných plošných zdrojů značně přispívá k realistickému zobrazení. V real-time aplikacích je tyto světelné efekty ve většině případů nutné nějakým způsobem předem vypočítat, či jejich efekt pouze přibližně napodobit.

2.1 Lightmapping

Lightmapping je technika, při které se vytvoří textury 3D objektů, které uchovávají informaci o nasvícení daných objektů. Lightmap textury jsou ve své podstatě stejné, jako textury obsahující informaci například o barvě objektu. Mají ale z pravidla daleko menší rozlišení. Lightmapové textury se vytváří po zasvícení virtuální scény většinou pomocí ray-tracingu. Tento proces probíhá offline, nikoli v reálném čase, díky čemuž je možné docílit vysoké kvality stínů, odraženého a rozptýleného světla.

Omezením této techniky je, že ji lze použít pouze pro statické objekty a statické nasvícení scény. Respektive pro animované objekty je to také možné, ale pro každý frame je nutné udělat lightmapu zvlášť. Tím však drasticky stoupají paměťové nároky.[23] Obrázek obr. 2.1-1 ukazuje, jak se promění vzhled objektu po nasvícení s využitím lightmapy.
[24]

obr. 2.1-1 Obr. Zobrazení jedné scény (1. Nenasvícené s texturou, 2. Nasvícené bez textury, 3. Výsledné zobrazení vzniklé spojením dvou předchozích) (Cadkid69 2020)

[23] CHANNA, K. Light mapping - Theory and amplementation. 2003.
[24] CADKID69. baked lightmap from blender. 2020.

2.2 Lightmapping pro dynamické objekty

Lightmapy, o kterých byla předchozí kapitola, se starají o realistické nasvícení pouze statických objektů. Nepřímé nasvícení pohyblivých objektů, jako například postav, však stejným způsobem řešit nelze. Metoda, která tento problém řeší a kterou využívá software Unreal Engine 4 se jmenuje Volumetric lightmaps. Princip, na kterém je tato metoda založena je ale podobný i u ostatních real-time aplikací. Metoda spočívá v rozmístění bodů v prostoru, ve kterých se předem vypočítá nepřímé osvětlení. Když se při běhu programu dynamický objekt pohne, je pomocí interpolace vypočteno nepřímé nasvícení tohoto objektu z bodů v prostoru, které jsou předem vypočítány. Výrazný rozdíl v realističnosti nasvícení postavy nepřímím světlem při použití Volumetric lightmap bez použití Volumetric lightmap můžeme vidět na obr. 2.2-1 a obr. 2.2-2.
[25]

obr. 2.2-1 Scéna s pohyblivým objektem bez použití Volumetric Lightmaps. (vlastní tvorba) (Sun tample demo, Unreal Engine 4)

obr. 2.2-2 Scéna s pohyblivým objektem s použitím Volumetric Lightmaps. (Sun tample demo, Unreal Engine 4)

[25] EPIC GAMES Sun Temple. In., 2014, vol. 2022.

Hustota bodů, ve kterých se předem vypočítá nepřímé osvětlení může být různá. Na jejím optimálním nastavení závisí jak vizuální vjem, tak výpočetní náročnost operace. Automaticky program rozmisťuje více bodů do blízkosti objektů a méně do volného prostoru, kde změny v odraženém světle již nejsou tak výrazné.
[26]

obr. 2.2-3 Scéna s vizualizací bodů v propstoru, ve kterých se vypočítává odražené světlo. (vlastní tvorba) (Sun tample demo, Unreal Engine 4)

[26] EPIC GAMES Sun Temple. In., 2014, vol. 2022.

2.3 Zastínění okolím (Ambient Occlusion)

Metoda Ambient Occlusion napodobuje chování světla v záhybech, hranách a místech kde na sebe jednotlivé části objektu navazují. Jde o způsob stínování objektu na základě výpočtu zastínění od okolních ploch. Jedná se o jeden ze způsobů, jak napodobit globální osvětlení bez využití realtime ray-tracingu.

Princip metody spočívá ve vysílání paprsků z bodů na 3D modelu všemi směry. Pokud paprsek nenarazí do okolních ploch, ale projde až k obrázku prostředí, respektive obloze, je tento bod tímto paprskem osvětlen. Pokud paprsek narazí do okolní plochy, pak tímto paprskem bod osvětlen není. Podle poměru paprsků, které narazí do okolních ploch a těch, které nenarazí se vypočítá, jak moc je daný bod zastíněn.[27]

Mapa Ambient Occlusion se obvykle vypočítává předem, při vytváření textur objektu, a proto nemůže reagovat na změny prostředí v reálném čase. Jde tedy jen o aproximaci, jak by se světlo na objektu chovalo ve většině situacích. Obvykle se používá s dalšími metodami simulace globálního osvětlení a spíše jako jemný efekt, který pocit realističnosti jen podtrhuje. Existuje ale i metoda zvaná Screen Space Ambient Occlusion, jejíž efekt se vypočítává v reálném čase pouze v zobrazované části scény. Její efekt je však jen velmi nevýrazný a má jisté zpoždění v zobrazení při rychlém pohybu prostorem.

Na obr. 2.3-1 vidíme model letounu se zobrazenou mapou Ambient Occlusion. Tmavá místa jsou plochy, které jsou nejvíce zastíněné okolím, zatímco světlá místa vidí téměř celé okolí a jsou tedy nejméně zastíněna.[28]
[29]

obr. 2.3-1 mapa Ambient Occlusion zobrazená na modelu B-35 Mitchell. (Kurachi 2007)

[27] MCREYNOLDS, T. Advanced Graphics Programming Using OpenGL. Elsevier Inc., 2005. ISBN-13: 978-1558606593

[28] KURACHI, N. The Magic of Computer Graphics. Edtion ed. Tokyo, Japan: Ohmsha, Ltd.,, 2007. ISBN ISBN: 978-1-56881-577-0.

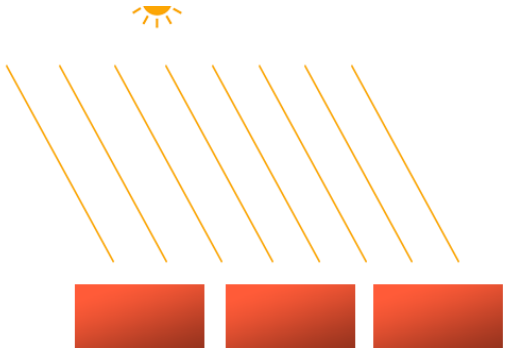
[29] KURACHI, N.

3 | Druhy světelných zdrojů a jejich parametry

Ve většině 3D softwarech se setkáme s pěti základními typy světelných zdrojů. Je běžné, že každý software má i své vlastní druhy světelných zdrojů, ty jsou však většinou jen již přednastaveným jedním ze základních světelných zdrojů určeným pro specifickou aplikaci. Těmito základními typy jsou directional light, point light, spot light a rectangle light a Skylight. Každý z těchto typů světelných zdrojů má parametry, kterými ovlivňujeme nejen jaké světlo zdroj emituje, ale také jak na toto světlo reagují objekty ve virtuální scéně. Zvlášť u realtime aplikací je velmi důležité myslet nejen na vizuální dojem scény ale i na výpočetní náročnost daného efektu. I když simulace osvětlení je navrhována tak, aby vycházela z fyzikálních zákonů, nebo je alespoň napodobovala, je nutné přistupovat k určitým ústupkům, či používat jiný přístup zesvětlování scény, než jaké bychom použili v reálném světe. Na druhou stranu, tyto omezení a možnost ovlivňovat nastavení zdrojů světla jiným způsobem, než v reálném prostředí dávají tvůrci i nové možnosti ve vytváření estetické kvality obrazu.

3.1 Directional light (směrové světlo)

Pokud je zdroj světla ve vzdálenosti, kterou můžeme považovat za nekonečnou, paprsky, které dopadají na objekty ve scéně jsou rovnoběžné. Rovnoběžnost paprsků bez závislosti na umístění objektů je hlavní definicí světelného zdroje directional light. Běžným příkladem takového světelného zdroje v reálném světě je slunce, které ve výpočtu nasvícené scény můžeme považovat za nekonečně vzdálený zdroj. Protože pozice objektu ve virtuální scéně nehraje vzhledem k paralelnosti paprsků roli, výpočet nasvícení takovým světlem bude pro všechny předměty podobný.[30] Světlo z tohoto zdroje také neztrácí na intenzitě v závislosti na vzdálenosti.
[31]



obr. 3.1-1 Znázornění zvětelného zdroje Directional Light a jeho paprsků dopadajících na objekty virtuální scény. (Vries, Basic Lighting nedatováno)

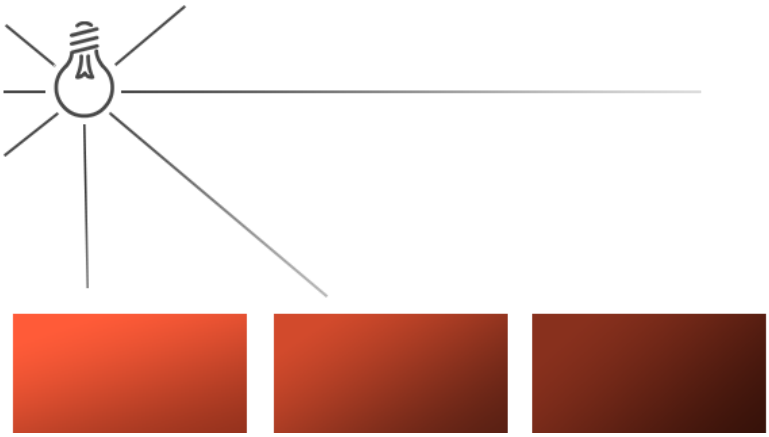
Directional light obvykle používáme k zasvícení celé scény, například exteriérů při simulaci přímého slunečního svitu. Lze také použít pro svícení interiérů světlem přicházejícím skrz okna. V takovém případě je ale dobré mít na paměti, že musíme daný program upozornit na to, že světlo, které má na interiérovou scénu efekt, přichází pouze oknem, tedy omezeným prostorem. Tím docílíme toho, že při výpočtu bude program soustřeďovat více paprsků pouze skrz okno a tím výrazně snížíme výpočetní náročnost a zároveň zvýšíme kvalitu výsledného efektu.

[30] VRIES, J. D. Learn OpenGL: Learn modern OpenGL graphics programming in a step-by-step fashion. Edtion ed.: Kendall & Welling, 2020. ISBN 9090332561.

[31] VRIES, J. D. Basic Lighting. In Learn OpenGL.

3.2 Point light (bodové světlo)

Point light neboli bodové světlo je světelný zdroj, který je umístěný v prostoru virtuální scény a svítí do všech směrů stejnou intenzitou. Světelné paprsky bodového zdroje, na rozdíl od paprsků vyzařovaných směrovým světlem, nejsou rovnoběžné a ztrácí intenzitu se vzdáleností od svého zdroje. Úbytek světla se vzdáleností můžeme u bodového zdroje nastavit. Tím se sice vzdalujeme od toho, jak se světlo chová v realitě, ale zároveň to tvůrci dává větší flexibilitu při vytváření požadovaného efektu. Nastavení úbytku ale není nezbytné a defaultně odpovídá tomu, jak světlo ubývá se vzdáleností ve skutečnosti. Nutné je však nastavit správně maximální dosah, kam paprsky bodového zdroje mohou doputovat. Tím nejenže může být dosaženo podobného efektu, jako nastavením úbytku světla se vzdáleností, ale je to důležité zejména pro zjednodušení výpočtu.
[32]

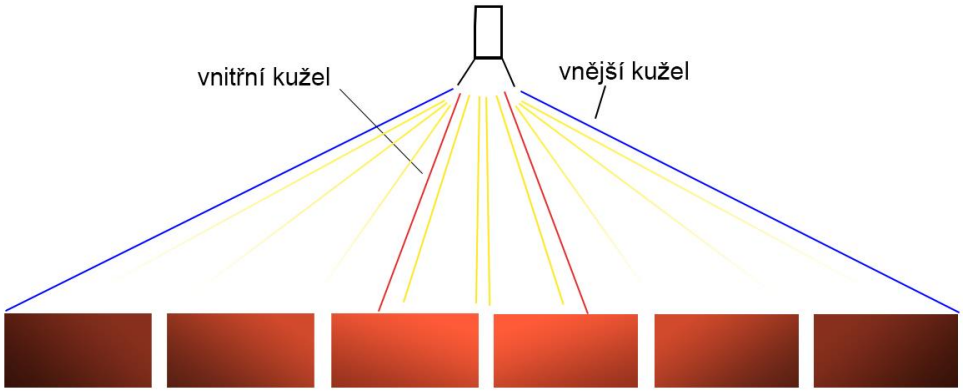


obr. 3.2-1 Znázornění bodového světelného zdroje a jeho paprsků dopadajících na objekty virtuální scény (Vries, Basic Lighting nedatováno)

[32] VRIES, J. D. Learn OpenGL: Learn modern OpenGL graphics programming in a step-by-step fashion. Edtion ed.: Kendall & Welling, 2020. ISBN 9090332561.

3.3 Spot light (reflektorové světlo)

Spot light neboli reflektorové světlo se chová principiálně stejně, jako bodové světlo, až na to, že nevyzařuje ve všech směrech, ale pouze v určitém prostorovém úhlu. Obvykle u takového světelného zdroje je možné nastavit dva prostorové úhly. Jeden, který udává velikost kuželu, ve kterém světlo vyzařuje maximální intenzitou a druhý, který určuje rychlost úbytku intenzity ve směru od středu kužele prvního.

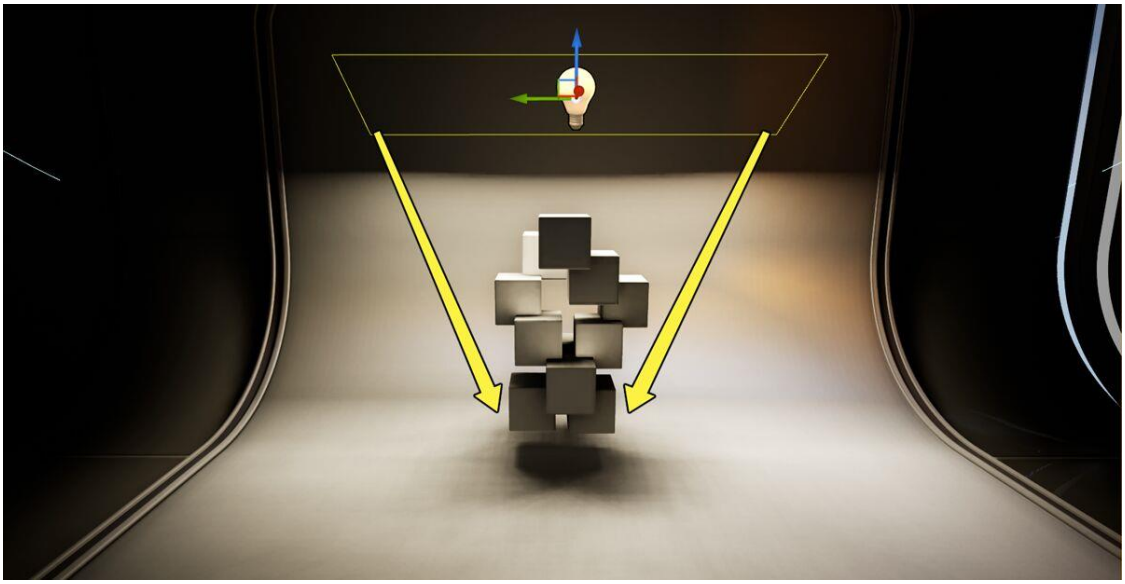


obr. 3.3-1 Znáyornění reflektorového světla a jeho paprsků dopadajících na objektz ve scéně (vlastní tvorba, Photoshop)

3.4 Rectangle light (plošné světlo)

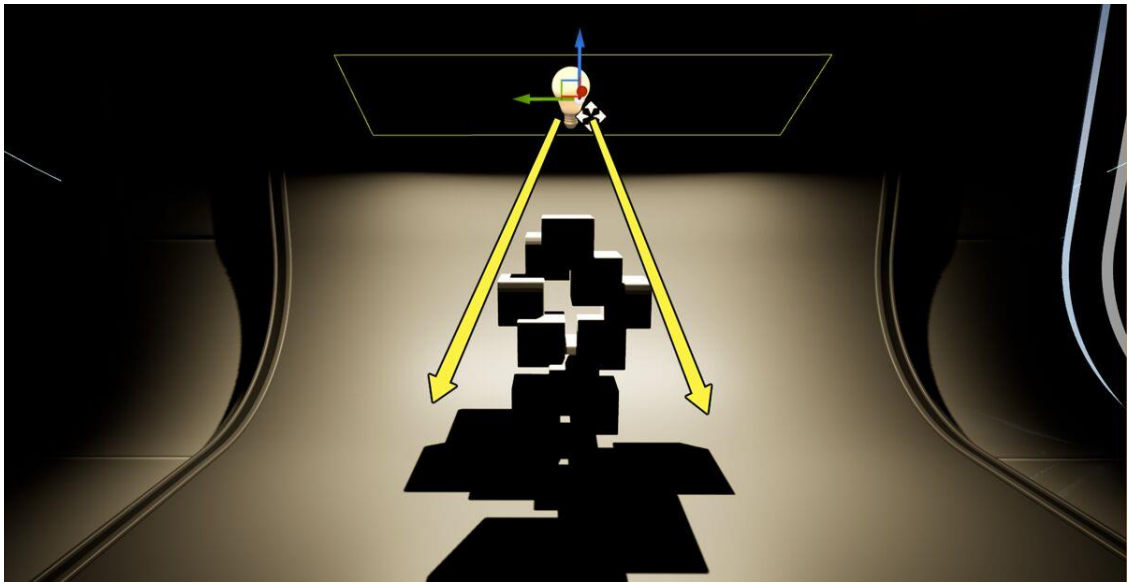
Plošné světlo vyzařuje paprsky z plochy o libovolně velkém obdélníku. Podobně jako u reflektorového světla můžeme nastavit vnější hranici, za kterou už světlo svítit nebude. Problémem tohoto typu světelného zdroje je, že v real-time aplikacích není obvykle možné ho využívat bez předchozího vytvoření lightmap například pomocí ray-tracingu. Při použití bez předvypočítaných lightmap se bude plošné světlo chovat podobně, jako bodové, nebo reflektorové světlo. Kvůli co možná nejjednoduššímu výpočtu je nutné, aby světlo z virtuálního světelného zdroje vycházelo z jediného bodu, nikoliv z plochy. Plocha světelného zdroje je v takovém případě použita pouze pro odrazy světelného zdroje v lesklých předmětech (specular light). Výsledkem toho jsou ostré stíny stejné, jako bychom získali při použité bodového zdroje, přestože v odlescích na předmětech uvidíme tvar plošného zdroje.

[33]



obr. 3.4-1 Plošný zdroj světla, který byl zahrnut do výpočtu lightmap. Stíny, které objekty vrhají jsou výrazně realističtější. (Rect Lights, 2020)

[34]



obr. 3.4-2 Plošný zdroj světla, který nebyl zahrnut do výpočtu lightmap. Stíny, které objekty vrhají jsou stejně ostré, jako kdyby byl zdroj světla bodový. (Rect Lights, 2020)

[33] EPIC GAMES, Rect Lights. In *unreal Engine 4.26 Documentation.* Epic Games, Inc.
[34] EPIC GAMES, Rect Lights. In *unreal Engine 4.26 Documentation.* Epic Games, Inc.

3.5 Skylight (světlo oblohy)

Pokud potřebujeme do tmavých částí scény dostat více rozptýleného světla a zároveň chceme, aby toto světlo bylo určitým způsobem modulované a nepřicházelo ze všech směrů rovnoměrně, můžeme použít světlo, které se ve většině 3D softwarech nazývá Skylight. Skylight je formou aproximace globálního osvětlení, které přichází ze vzdálených objektů jako je obloha. Můžeme ho však využít i pro aproximaci světla přicházejícího například od nasvícených budov nebo i jedné místnosti. Jinými slovy pro simulaci odrazů od objektů obklopující naši scénu. Skylight funguje tak, že zachytí barevnost a světlost svého okolí ve vzdálenosti, které mu určíme, a to aplikuje v podobě světla na vytvářenou scénu.

K zachycení svého okolí používá Skylight tzv. cubemapu neboli HDRi (High dynamic range image). Ten může být buď vytvořen přímo z okolí scény, kterou vytváříme, nebo může být použita jakákoli vlastní HDR fotografie. Díky tomu, že je fotografie zachycena ve velkém dynamickém rozsahu, je pro program snáze možné zjistit, kde se nachází zdroj světla viz obr. 3.5-1.

[35]

obr. 3.5-1 HDRi image v různých stupních expozice. (Brais Brenlla Ramos 2019)

Pokud Skylight používáme jako dynamický zdroj osvětlení je potřeba brát v potaz, že světlo z něho vytvořené je aplikováno na všechny objekty ve scéně. Pokud například Skylight simuluje světlo přicházející z oblohy, je toto světlo aplikováno na všechny objekty bez ohledu na to, jestli

[35] RAMOS, B. B., DORAN, J. P. Unreal Engine 4 Shaders and Effects Cookbook: Over 70 recipes for mastering post-processing effects and advanced shading techniques. Packt Publishing, 2019. ISBN 1789538548.

je objekt v exteriéru nebo interiéru. To lze různými způsoby vyřešit, například vytvořením oblasti, ve které Skylight nepůsobí. Skylight může být použit i jako statický zdroj a pak je jeho efekt vypočítán pomocí ray-tracingu a přidán do lightmap. V tomto případě je světlo z něho zastíněno objekty stejně, jako se to děje u ostatních světel.

4 | Parametry světelných zdrojů

4.1 Light mobility

V real-time aplikacích lze využívat několik druhů nasvícení scény. Dynamické nasvícení, statické nasvícení a jejich kombinaci, tzv. stacionární nasvícení. Každý z těchto nastavení mobility světelných zdrojů dosahuje rozdílných vizuálních výsledků a má značně rozdílné nároky na výpočetní výkon při běhu programu, protože může určovat jakým způsobem se obraz renderuje. Nastavení těchto typů lze provádět pro každý světelný zdroj zvlášť a tím dosáhnout vhodné rovnováhy mezi výpočetní náročností a požadovaným vizuálním efektem.

Základním rozdílem mezi statickým a dynamickým nasvícením je, zda se pro daný světelný zdroj předem vypočítávají lightmapy či nikoliv. Principu fungování Lightmap se věnuje kapitola 2.1. Lightmapy sice vytvářejí velmi dobré výsledky v simulaci osvětlení díky globální iluminaci, ale zároveň přináší velká omezení při pohybu objektů, či samotných světelných zdrojů v prostoru. Stejně jako není z technického hlediska výhodné používat lightmapy pro pohybující se objekty, tak není výhodné, ani většinou kvůli nárokům na grafickou paměť možné, aby byly lightmapy použity v případě pohybujících se světelných zdrojů. Princip tohoto problému je stejný v obou zmíněných případech. Pro každý frame vzájemné změny objektu a světelného zdroje by musela být předem vypočítána nová sada lightmap, což by vedlo k zahlcení pamětí.

Pohyblivá světla je možné u real-time aplikacích použít pouze bez předpočítaných lightmap, tedy jako dynamické nasvícení, ať už pomocí rasterizace nebo některé formy ray-tracingu. Proto je obvykle nutné přístupy kombinovat. S novými technologiemi, které napodobují globální osvětlení bez nutnosti vytvářet lightmapy, a se zvyšujícím se výkonem osobních počítačů, se však postupně čistě dynamické nasvícení scény stává vizuálně velmi konkurenceschopnou alternativou, která dává tvůrci větší volnost.

Software pro vytváření realtime aplikací obvykle nabízí tři základní nastavení mobility světelných zdrojů. statické, stacionární a pohyblivé.

4.1.1 Static (statické)

Statické světlo je světelný zdroj, se kterým nelze pohybovat ani jinak měnit jeho nastavení při běhu programu. Statické světlo nemá žádný efekt na pohyblivé objekty. Ani je neosvětluje, ani od statického světla pohyblivé objekty nevytváří stíny. Statické světlo je předem vypočítáno pouze do lightmap statických objektů. Díky tomu má nulový dopad na výpočetní zátěž při běhu programu. Je ale nutné počítat s časově náročným procesem výpočtu renderování lightmap,

které je ve většině případů realizováno pomocí ray-tracingu. Statické světlo simuluje reálné osvětlení nejvěrněji, a to zejména v případě plošných zdrojů, ale pouze na statických objektech.[36]

Vhodné je tedy v situacích, kdy je jeho efekt omezen na část scény, kde nedochází ke vzájemné změně polohy objektů a zdroje světla, či v případech, kdy je cílem snížit výpočetní náročnost programu, jako například u aplikací pro mobilní zařízení.

4.1.2 Movable (dynamické)

Protože dynamický světelný zdroj není při renderování lightmap brán vůbec v potaz, je jeho polohu v prostoru i další nastavení možné jakkoli měnit při běhu programu. Díky tomu je plně reaguje na pohyblivé objekty ve scéně. Velkou nevýhodou dynamického světelného zdroje je, že bez použití speciálních technik jako je ray-tracing nebo Screen Space Global Illumination nevytváří žádné globální osvětlení a vyžaduje ze všech nastavení mobility světelných zdrojů největší výpočetní výkon.

Použít dynamický světelný zdroj je nutné v situacích, kdy je vyžadován pohyb zdroje světla ve virtuálním prostoru. Důležité je pamatovat na velikost plochy, na kterou má světlo mít efekt. Čím je tato plocha větší, respektive čím více objektů obsahuje, tím větší bude dopad na výpočetní náročnost.[37] (mobility nedatováno)

4.1.3 Stationary (stacionární)

Stacionární světelný zdroj kombinuje některé vlastnosti obou předchozích typů. Stacionární světelný zdroj se nemůže hýbat, ale může měnit své ostatní parametry, jako je intenzita nebo barva. Na rozdíl od statického světa má stacionární zdroj vliv na pohyblivé objekty. Přímé světlo (lokální osvětlení), které tento zdroj vytváří se vypočítává při běhu programu a díky tomu se mohou měnit jeho parametry. Globální osvětlení se ukládá do textur pomocí lightmap, které už při běhu programu měnit nelze. Proto není možné světlem pohybovat, a proto se také například změna intenzity světla projeví pouze na lokálním osvětlení, nikoliv globálním.[38]

Jedná se o nejčastěji používaný typ osvětlení ve většině situací, protože dosahuje dobrých vizuálních výsledků, určitých možností interaktivity při běhu programu a přijatelné výpočetní náročnosti.

[36] EPIC GAMES, Static Lights. In *Unreal Engine Documentation.*
[37] EPIC GAMES, Mobility, Movable Lights.
[38] EPIC GAMES, Mobility, Static Lights.

5 | Stíny v real-time aplikacích

Jednou z nejdůležitějších složek tvořících iluzi prostoru ve dvoudimenzionálním obraze jsou stíny. Podle stínů divák lépe rozpoznává tvary objektů, jejich vzájemné vzdálenosti a směr, ze kterého přichází světlo. Stíny výrazně přispívají k celkové realističnosti obrazu.

Stíny jsou taky velmi silný stylizační prvek, kterým tvůrce může podpořit atmosféru scény, její vyznění a dosáhnout požadované výtvarné podoby. V reálném světě jde světlo a stín ruku v ruce. Kameraman vybírá světelné zdroje s ohledem na kvalitu světla a má i jasnou představu o vzhledu stínů, které budou snímané objekty nasvícené daným světlem vrhat. Ve virtuálním prostoru real-time aplikací musí ale tvůrce k světelným zdrojům a stínům přistupovat více odděleně. Není pravidlem, že by daný zdroj světla musel vrhat stíny vždy stejným způsobem. Tvůrce může ovlivňovat jejich kvalitu jako například ostrost stínů, jak hluboké budou, zda jejich tvar bude přirozený, nebo nějakým způsobem deformovaný, nebo jestli objekt vůbec stíny vrhá či nikoliv.

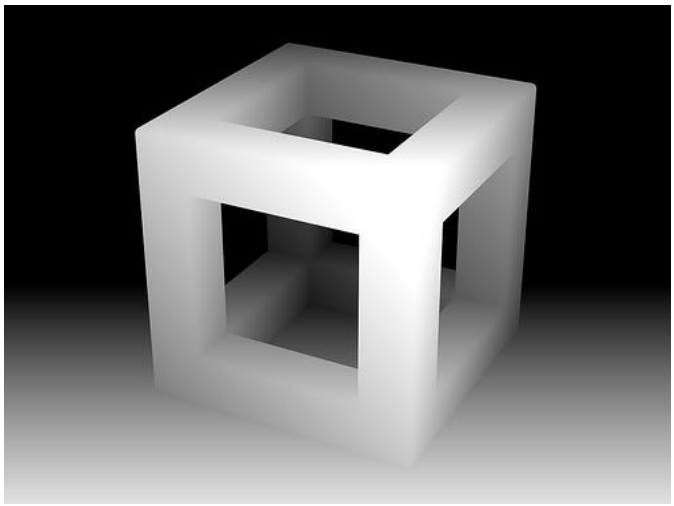
Z fyzikálního hlediska je stín definován jako oblast v prostoru s absencí světla, která je způsobená blokací světelných paprsků.[39] Pro výpočet stínů tedy musíme zjistit, zda existuje přímka z daného místa ke zdroji světla, která není přerušená jiným objektem. To je zjednodušeně také princip, na kterém funguje ray-tracing. V real-time aplikacích používajících rasterizaci však existují výpočetně méně náročné metody. Ty můžeme rozdělit podle techniky, jakou jsou stíny generovány do několika kategorií.

[39] CAMBRIDGE DICTIONARY. Meaning of shadow in English.

5.1 Shadow Mapping

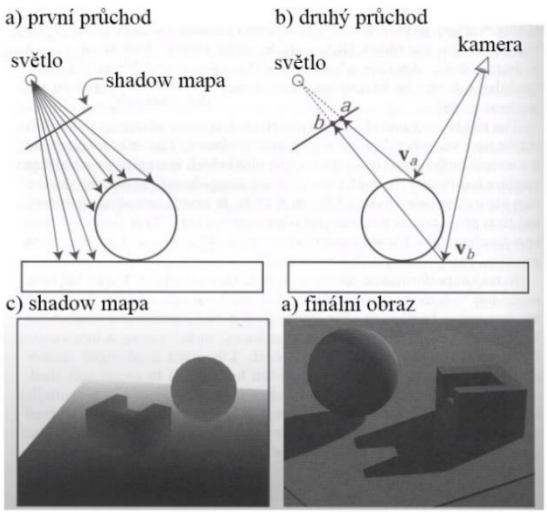
Metoda Shadow mapping zjišťuje, zda je daný pixel ve stínu či nikoliv podle toho, zda je přímo viditelný z pozice světelného zdroje. Nejprve je scéna vykreslena z pozice světla podobně jako při zobrazení virtuální kamerou pomocí rasterizace. Do vzniklé textury se pomocí z-bufferu uloží informace o hloubce každého pixelu. Této textuře se říká Shadow mapa. Poté je scéna vykreslena z pozice virtuální kamery se informace o hloubce ze z-bufferu obou vykreslení porovnají. Body ve scéně, které jsou ve stejném místě, ale jejichž informace o hloubce z obou vykreslení neodpovídají, jsou z pohledu zdroje světla zakryty jiným objektem, a tak jsou vykresleny jako ve stínu.

[40]



obr. 5.1-2 Depth mapa. Informace o hloubce vykreslená z pozice kamery. (company 2019)

[41]



obr. 5.1-1 Jednotlivé kroky při zobrazování stínů pomocí techniky shadow mapping (James D. Foley 1995)

Pokud je ve scéně více světel, je potřeba mít pro každé z nich svou vlastní shadow mapu a při finálním vykreslení z pozice kamery porovnávat informace o hloubce s každou shadow mapou zvlášť. Při vykreslování depth map, tedy informací o hloubce je možné zcela přeskočit veškeré grafické operace, které neovlivňují siluetu objektu (barvy, normály, odrazivost objektu, hrubost). To může výsledný čas výpočtu výrazně snížit. I přesto je potřeba myslet na to, že každý další světelný zdroj bude zpomalovat výpočet a potencionálně ovlivňovat snímkovou frekvenci.[42]

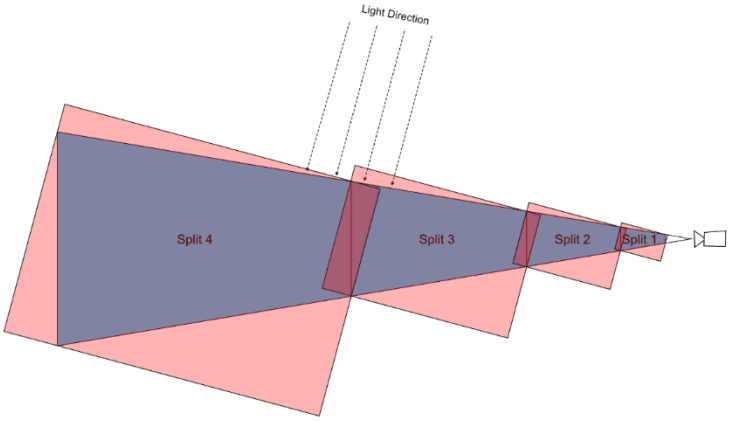
[40] COMPANY, S. NULL. Reflective Shadow Maps: Part 1. 2019.

[41] FOLEY J. D., VAN DAM A., FEINER S K., HUGHES, J. Computer Graphics: Principles and Practice. United States: Addison-Wesley, 1995. ISBN 978-0321399526.

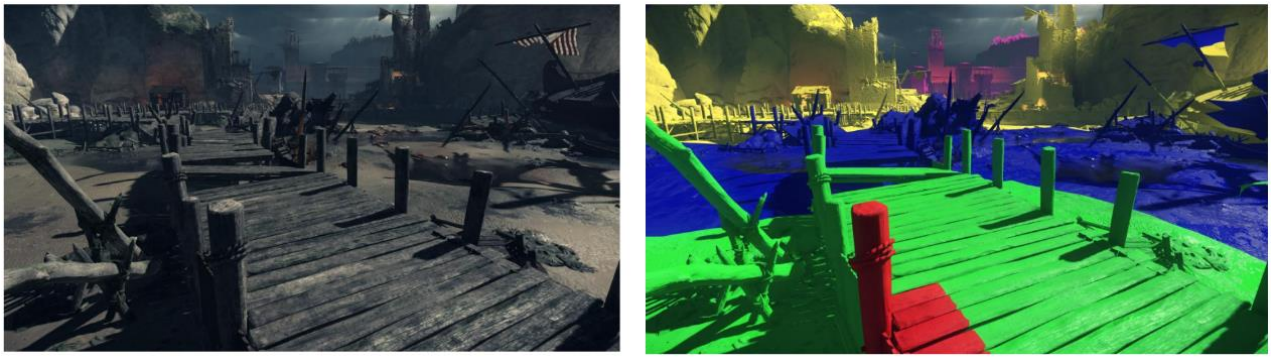
[42] ROHÁČEK, D. Improving probes in dynamic diffuse global illumination Vysoké Učení Technické v Brně, 2018.

5.2 Cascaded Shadow mapping

Cascaded Shadow mapping je technika, která vylepšuje metodu Shadow mapping. Pokud se snažíme pomocí techniky Shadow mapping generovat stíny na velkém území, například v rozsáhlém exteriéru nasvíceném sluncem, musela by mít Shadow mapa neprakticky velké rozlišení, aby byly stíny vizuálně přijatelné jak v blízkosti virtuální kamery, tak daleko o ní. To je v praxi často nedosažitelné. Tento problém řeší metoda Cascaded Shadow mapping tak, že pro každé světlo vytvoří sadu několika shadow map, většinou tři až pět, které pokrývají různou oblast v závislosti na vzdálenosti od virtuální kamery. Díky tomu mohou být stíny vrhané od objektů blízko kamery velmi kvalitní, zatímco ty v dálce mohou svou kvalitu postupně ztrácet. I když tato metoda je výpočetně náročnější než klasický Shadow mapping, oproti použití jedné shadow mapy o velkém rozlišení pro celou scénu je výrazně úspornější.[43]
[44]



obr. 5.2-1 Znázornění tvorby Cascaded shadow mapping. (Leach
[45]



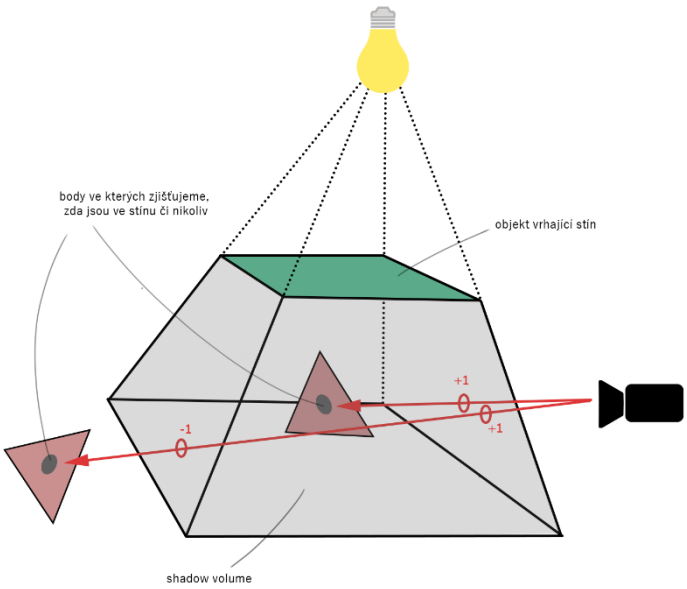
obr. 5.2-2 Vlevo: výsledný obrázek s použitím Cascaded Shadow mapping. Vpravo: Vizualizace jednotlivých Shadow map v závislosti na vzdálenosti od kamery (Kasyan 2013)

[43] WHITE S, COULTER V. N. D., JACOBS M., SATRAN M. Cascaded Shadow Maps. In.: Microsoft, 2020, vol. 2022, p. 1.
[44] LEACH, C. failengine. In Craig Leach, Software Developer. Craig Leach, 2019.
[45] KASYAN, N. Playing with Real-Time Shadows. 2013.

5.3 Shadow Volumes

Metoda výpočtu stínů v real-time aplikacích Shadow Volumes se ve své podstatě velmi jednoduchá a dosahuje velmi dobrých vizuálních výsledků s minimálními artefakty oproti technice Shadowmaps. I přesto není příliš používána, zejména pro její vysokou výpočetní náročnost při určitých typech scén.

Princip této metody spočívá v zjišťování kolikrát a jakým způsobem paprsek vyslaný z virtuální kamery na daný bod v prostoru projde přes tzv. shadow volume. Shadow volume je objekt, který reprezentuje prostor, který je v zákrytu nějakého objektu z pohledu zdroje světla. Představuje tedy prostor, v němž jsou všechny body ve stínu. Pokud je vyslán paprsek z virtuální kamery směrem k bodu, o kterém chceme zjistit, zda je ve stínu či nikoliv, počítáme, zda a kolikrát vstoupí a vystoupí z Shadow volume. Když zaznamenáme průchod do Shadow volume přičteme 1, když zaznamenáme vystoupení z Shadow volume odečteme 1. Pokud je výsledkem tohoto výpočtu nula, je daný bod daným světlem osvětlený. Pokud je výsledkem jiné číslo, je daný bod ve stínu viz. obr. 5.3-1 .[46]
[47]

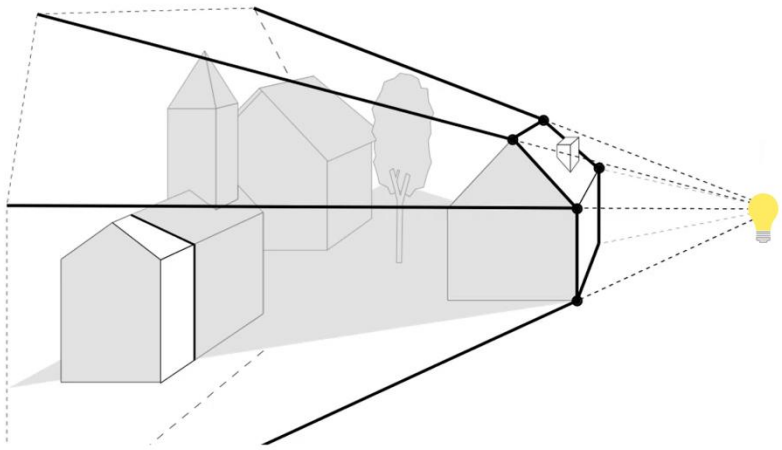


obr. 5.3-1 Základní princip fungování Shadow Volumes. Pro každý bod, u kterého chceme zjistit, zda je ve stínu či nikoli, počítáme průchody paprsku přes shadow volume od kamery k tomuto bodu. (Solomon 2020)

Výhodou této metody je, že nedochází k aliasingu, jako je to v případě techniky Shadow mapping, protože paprsků můžeme vyslat přesně tolik, kolik chceme zobrazit bodů, nezávisle na jejich umístění v prostoru. Rozlišení stínů zobrazených na obrazovce díky tomu není závislé na vzdálenosti kamery od daného předmětu. Velkým problémem však může být vysoká výpočetní náročnost, která rychle stoupá s množstvím polygonů a množstvím světelných zdrojů. Částečně

[46] SOLOMON, J. Introduction to Computer Graphics. In., 2020.
[47] SOLOMON, J. Introduction to Computer Graphics. In., 2020.

je možné zjednodušit výpočet tak, že se shadow volume bude vytvářet pouze ze zjednodušených verzí objektů. Viz. obr. 5.3-2. Vynechají se detaily objektu, které nejsou pro vzhled stínu příliš významné. Pokud bychom chtěli zobrazit touto metodou přesné stíny, které vrhá například listnatý strom, musíme pro každý list, větev a každý jiný detail vytvořit vlastní shadow volume, čímž se stává tato metoda velmi náročná a proto nevýhodná.
[48]



obr. 5.3-2 Znázornění principu Shadow volumes v komplexní scéně. (Solomon 2020)

5.4 Ray-traced Shadows

Dalším a nejpokročilejším způsobem je generování stínů pomocí ray-tracingu. Jak už bylo zmíněno v kapitole 1.5, Ray-tracing se v real-time aplikacích používá k vykreslování pouze některých efektů, které jsou součástí obrazu, a to kvůli přílišné náročnosti na výpočetní výkon. Jedním z těchto elementů jsou stíny.

Nejčastěji používaná metoda generování stínů (Cascaded Shadow Mapping) má oproti generování stínů pomocí ray-tracingu několik výrazných nevýhod. Dochází k perspektivnímu aliasingu, může vzniknout tzv. Peter Panning (kvůli nedostatečnému rozlišení Z-bufferu stíny nedosahují až k objektům, které je vytvářejí a proto vypadají, jako by levitovaly.), není možné generovat měkké stíny a stíny vytvářené polopropustnými materiály. I když bylo vyvinuto mnoho algoritmů pro potlačení těchto nedostatků, většinou je potřeba mnoho manuálního ladění pro každou scénu zvlášť.
Ray-tracing tak nabízí poměrně jednoduše aplikovatelné řešení bez nutnosti výrazného dalších úprav a s možností aplikace na libovolnou scénu. Jeho hlavním úskalím je však vysoká výpočetní náročnost, která se rychle zvyšuje s množstvím použitých světel, případě použitím plošných světel. A tak i s použitím hybridních renderovacích systémů, které kombinují rasterizaci a

[48] SOLOMON, J. Introduction to Computer Graphics. In., 2020.

ray-tracing je potřeba na hardwarová omezení stále myslet.[49]

Ostré stíny dokáže ray-tracing vykreslit velmi rychle a snadno tak, že z daného místa vyšle paprsek směrem ke světelnému zdroji a zkoumá, zda ho paprsek dosáhl či nikoliv. Zcela ostré stíny však v reálném světě nikdy nenajdeme, protože všechny zdroje mají určitou velikost. Zcela ostrý stín by vytvářel pouze světelný zdroj nekonečně malý, nebo nekonečně daleko. Každý světelný zdroj proto vytváří jak stíny, tak polostíny neboli přechod mezi zastíněným a nezastíněným. Ostré stíny vytvořené ray-tracingem tak budou vypadat realisticky pouze v případě zdroje osvětlení jako je například slunce, které můžeme považovat za nekonečně vzdálený světelný zdroj.

Pokud chceme pomocí ray-tracingu vykreslit měkké stíny bezchybně, museli bychom vyslat jeden paprsek do každého bodu světelného zdroje, což je v praxi nemožné, protože takových bodů je nekonečno. Základním principem vykreslování měkkých stínů pomocí ray-tracingu je vyslat jen velmi omezený počet paprsků do náhodných bodů na zdroji světla. Tak vznikne velmi zrnitý obraz stínu, které je možné různými metodami filtrovat a získat tak realisticky vypadající měkký stín.[50]

[49] J BOKSANSKY J., WIMMER M., BITTNER J. Ray Traced Shadows: Maintaining Real-Time Frame Rates. In *Ray Tracing Gems.* Prague: Czech Technical University in Prague, 2019.

[50] WESTER, A. Ray-tracing soft shadows in real-time. 2020.

II. Praktická část – reprodukce scény ve videoherním vývojovém prostředí

6 | Cíle praktické části

V rámci praktické části mé diplomové práce jsem vytvořil několik scén, které jsou co nejpřesnější reprodukcí existující fotografie nebo screenshotu z filmu. Každá z těchto scén je vytvořena v několika verzích pomocí různých technik. Výsledky dosažené těmito technikami porovnávám mezi sebou i s referenční fotografií. Reference není důležitá pro srovnání, jak blízko je možné přiblížit obraz ve virtuální realitě skutečnosti. Avšak hlavně pro to, abychom si uvědomili, jak by se světlo v prostředí dané scény chovalo v realitě. Díky tomu můžeme lépe hodnotit jevy, jako je primární a sekundární osvětlení, chování stínů, nebo reakce světla na různé povrchy. Hlavní techniky, které porovnávám jsou ray-tracing, statické osvětlení (osvětlení, které používá pouze světlo aplikované do textur) a dynamické osvětlení (simulace osvětlení vytvořená výhradně pomocí rasterizace). Zkoumám ale i důležité metody, které se při zasvětlování používají, jako jsou Skylight a Volumetric Lightmap. Důležitou součástí je také porovnání dosažených obnovovacích frekvencí. Díky srovnání kvality obrazového výstupu a povědomí o jejich výpočetní náročnosti si lze udělat představu o tom, zda je vhodné tyto techniky v dané scéně použít.

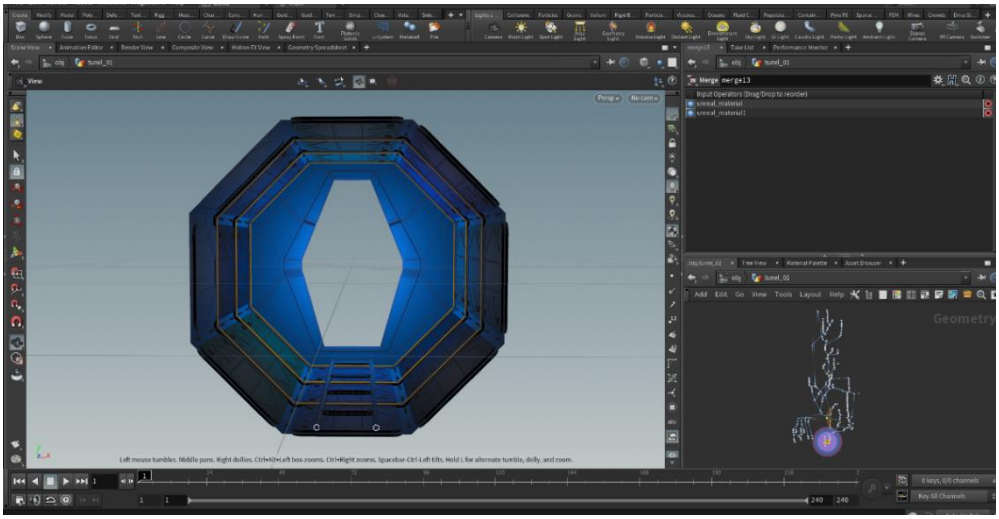
7 | Scéna 1 - 2001: A Space Odyssey

Ve scéně 1 jsem vytvořil prostředí, které se co nejblíže podobá dekoraci vesmírné lodi z filmu 2001: A Space Odyssey. Tuto scénu jsem vybral kvůli poměrně snadné identifikaci zdrojů světla, a tak možnosti přímo porovnat výsledky dosažené ve virtuálním prostoru s originálním záznamem bez přílišné odchylky v rozmístění a typu použitých světel. Protože je tato scéna poměrně jednoduchá ve smyslu počtu rekvizit, komplexnosti jejich tvarů, materiálů a textur, jsou na výsledku dobře patrné i drobné rozdíly, které by se snadno ztratily ve vizuálně komplexnější scéně.

7.1 Způsob vytváření scény ve virtuálním prostředí

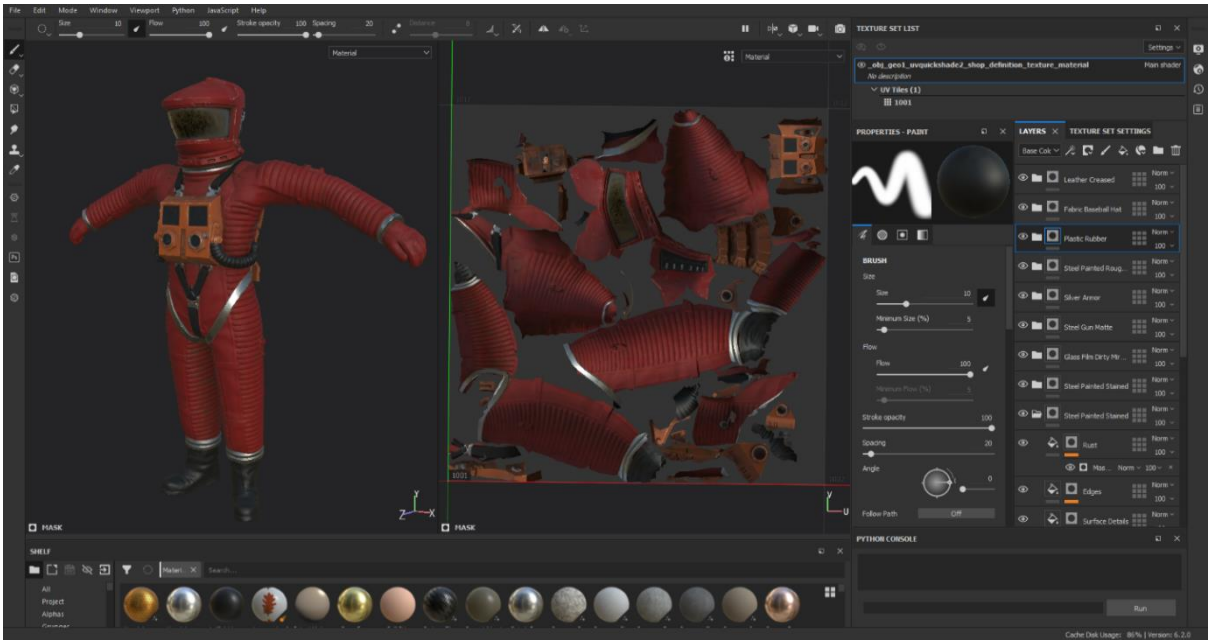
Model chodby vesmírné lodi, stejně jako modelu postavy astronauta jsem vytvořil na základě snímků z filmu. K dispozici jsem neměl technické výkresy dekorace, ani jinou detailnější dokumentaci ohledně použitých materiálů. Podobnost scén je tedy pouze přibližná a odchylky například v proporcích jednotlivých objektů jsou mohou být různé. Stejně tak jsem vycházel pouze z předpokladu, že scéna byla primárně zasvícená pomocí světelných panelů, které jsou přímo viditelné v obraze.

Všechny modely scény byly vytvořené v softwaru Houdini 18.5 a Zbrush 2020. Textury v softwaru Substance painter. Pro sestavení finální scény a její zasvícení byl použit software Unreal Engine 4.25.4.



obr. 7.1-1 Finální model bez textur v softwaru Houdini připravený pro export do software Unreal Engine (vlastní tvorba)

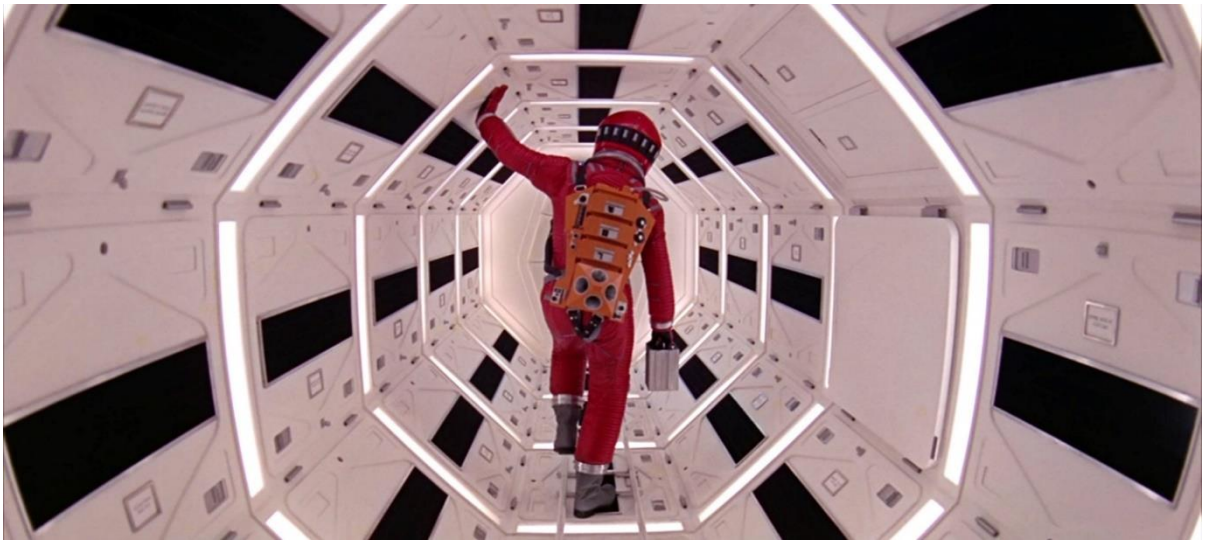
obr. 7.1-3 Nenasvícená scéna sestavená v softwaru Unreal Engine 4.25.4. (vlastní tvorba)



obr. 7.1-2 Otexturovaný model astronauta v softwaru Substance Painter. (vlastní tvorba)

7.2 Popsání principů použitých při zasvětlení vybrané scény

Z filmu 2001: A Space Odyssey jsem vybral jeden snímek, jako finální referenci, které jsem se z hlediska svícení, kompozice a tonálního podání snažil přiblížit.
[51]



obr. 7.2-1 Snímek z filmu 2001 a space odyssey použitý jako hlavní reference. (Kubrick, 2001: A Space Odyssey, 1968)

Prostor vybrané scény ve filmu je patrně zasvětlen pouze světelnými panely, které jsou přímo viditelné v obraze. Je možné, že v prostoru za kamerou jsou světelné zdroje v podobném rozložení, jako vidíme v záběru. Tento předpoklad vychází ze skutečnosti, že osvětlení stěn je rovnoměrné i v blízkosti okrajů obrazu.

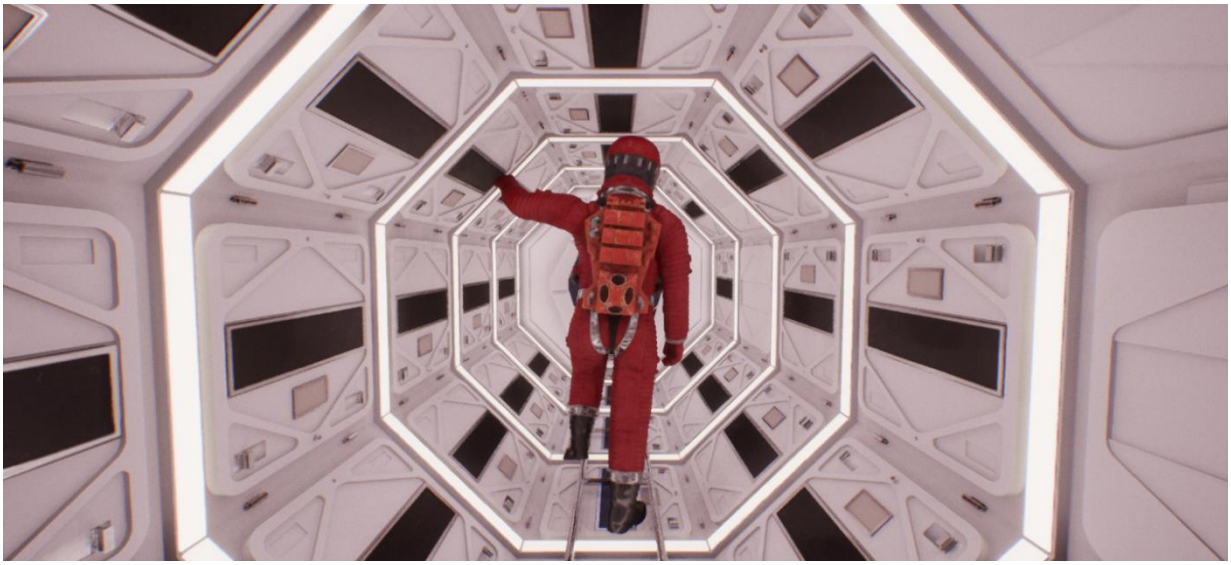
Rozmístění světel ve virtuálním prostoru je stejné, jako v reálném záběru. Světelné zdroje mají stejný rozměr, jako světelné panely na obvodu stěn. Použity byly zdroje plošné tzv. Rect Light. Všechny zdroje vyzařují stejnou intenzitou o stejné teplotě chromatičnosti. Mírný purpurový tón byl přidán při postprocesingu obrazu.

Při zesvětlování této scény ve virtuálním prostředí jsem použil tři různé přístupy, na kterých zkoumám jimi dosažené výsledky a porovnávám je se záběry z filmu a také popisuji jejich další použitelnost v realtime aplikaci, jako například ve videohře. Tyto tři přístupy dále nazývám Statická světla, Dynamická světla (rasterizace) a ray-tracing.

[51] KUBRICK, S. 2001: A Space Odyssey, Stanley Kubrick Productions, Great Britain, 1968

7.2.1 Statická světla

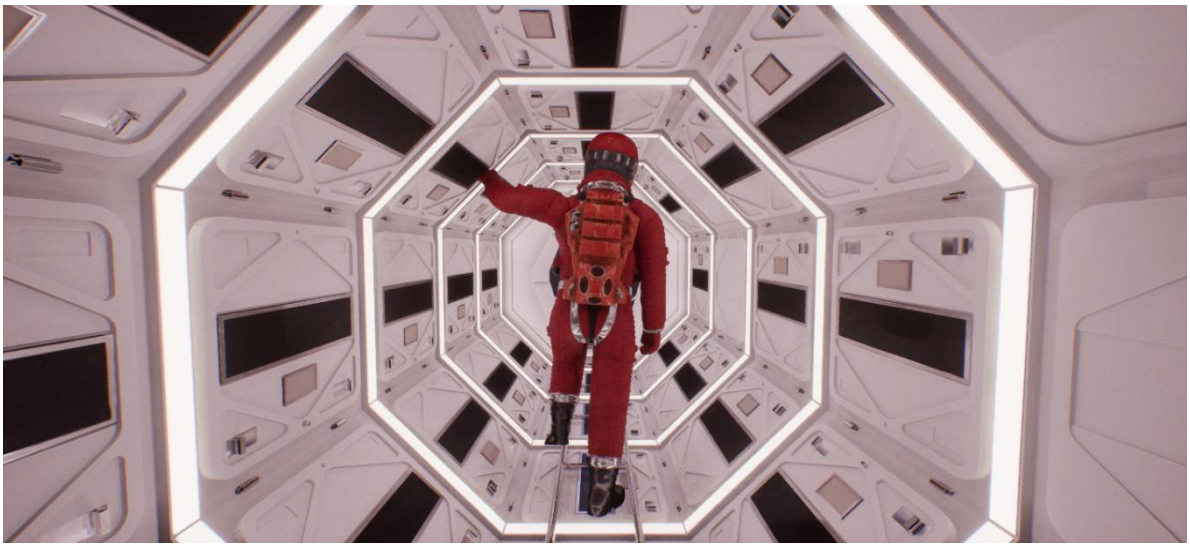
V tomto přístupu k zasvětlení scény jsem použil pouze statická světla, tedy světla, kterými nelze pohybovat při běhu programu, ani je nijak jinak měnit. Světelný efekt těchto světel se renderuje do textur, tzv. lightmap. Statická světla přímo neinteragují s pohyblivými objekty, tedy s postavou astronauta, která kvůli tomu nemůže vrhat žádné stíny a ani být přímo nasvícená těmito světelnými zdroji. Absence stínů není v tomto konkrétním případě problémem, protože rovnoměrné nasvícení z velkého množství světelných zdrojů stíny téměř eliminuje. Nasvětlení astronauta bylo nutné řešit pomocí lightmap pro dynamické objekty. Aproximace intenzity osvětlení v prostoru dokáže simulovat světelné rozdíly v různých částech prostoru, ale její rozlišení není dostatečné pro modulaci světla na detailech skafandru. Velkou výhodou tohoto přístupu je velmi nízká výpočetní náročnost při běhu programu. Při konfiguraci běžného pracovního počítače, byla obnovovací frekvence scény 120fps, což odpovídá maximální hodnotě, kterou dovoluje software Unreal Engine 4.



obr. 7.2-2 Zasvícená scéna s použitím statických světel. Unreal Engine 4.25.4, (vlastní tvorba).

7.2.2 Dynamická světla

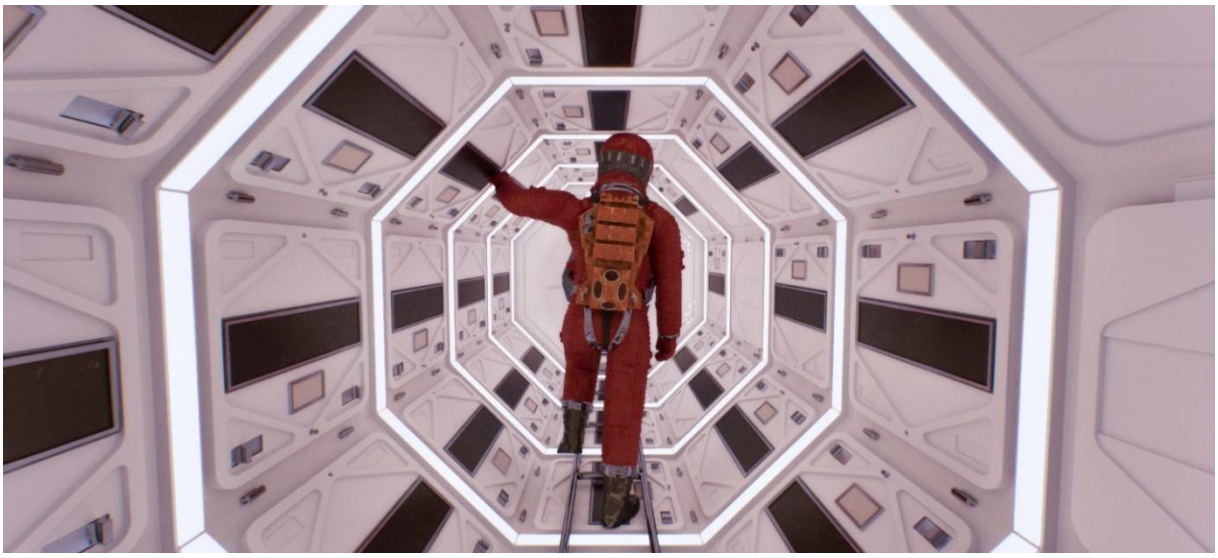
Velká výhoda použití dynamických světel spočívá v absenci nutnosti vytvářet UV Lightmapy pro každý objekt. Celkově se tím zkrátí doba výroby modelů a výrazně se zjednoduší proces zesvětlování scény. Výhodou dynamického svícení také je, že jak světla, tak objekty se mohou pohybovat, měnit intenzitu, barvu a další parametry při běhu programu. Nevýhodou pak je vizuálně méně přesné, nebo žádné sekundární osvětlení. Hlavním úskalím této metody je však její velká výpočetní náročnost. V porovnání se statickými světly „stojí“ dynamická světla až dvacetkrát více výpočetního výkonu. Při použití v testované scéně tak klesnula obnovovací frekvence z 120fps při použití statických světel na 25fps při použití dynamických světel. I když se jedná o nestandartní případ použití dynamických světel z hlediska jejich počtu, lze na něm dobře demonstrovat jejich výhody a nevýhody.



obr. 7.2-3 Zasvícená scéna s použitím dynamických světel. Unreal Engine 4.25.4, (vlastní tvorba).

7.2.3 Ray Tracing

Ray-tracing v této scéně byl použit nejen pro zobrazení odlesků, ale i přímého a sekundárního osvětlení. Jinými slovy, ray-tracing byl v tomto případě použit pro simulaci celého osvětlení scény všemi světelnými zdroji. Mělo by se tak jednat z hlediska věrnosti simulace osvětlení o nejpřesnější reprodukci. I když v real-time aplikacích by takovýto přístup nebyl vhodný z důvodu extrémní náročnosti výpočtu, zařadil jsem ho kvůli srovnání dosažených výsledků s metodami konvenčními. Obnovovací frekvence se v případě použití ray-tracingu pohybovala okolo jednoho snímku za 3 sekundy.



obr. 7.2-4 Zasvícená scéna s použitím ray-tracingu. Unreal Engine 4.25.4, (vlastní tvorba).

8 | Scéna 2 – Utah Monolith

Ve druhé scéně jsem se zreprodukoval fotografii zachycující skalnatý kaňon, ve kterém je umístěn kovový lesklý monolit. Kombinace ostrého denního světla přímo dopadajícího na matné skály i na lesklý povrch monolitu a odraženého světla od obou povrchů dává dobrou příležitost k porovnání chování světla v reálném prostředí a v prostředí virtuálním. Protože při pořízení originální fotografie nebylo použito žádné umělé světlo, pro vytvoření adekvátního virtuálního světla stačilo odhadnout pozici slunce podle tvaru a úhlu stínů. Při zkoumání rozdílů vzniklých renderů a originální fotografie byl hlavní zájem na globálním osvětlení, tedy osvětlení prostoru způsobeném odrazy od okolních předmětů, a na rozdílu v chování globálního světlení po odrazu světla od různých druhů materiálu.
[52]

obr. 7.2-1 Referenční snímek pro vytváření scény Utah Monolith (DAVE KOCH 2020)

[52] KOCH, D. The Problem with the Utah Monolith, 2020.

8.1 Popis způsobu vytváření reprodukce reálné scény ve virtuálním prostředí

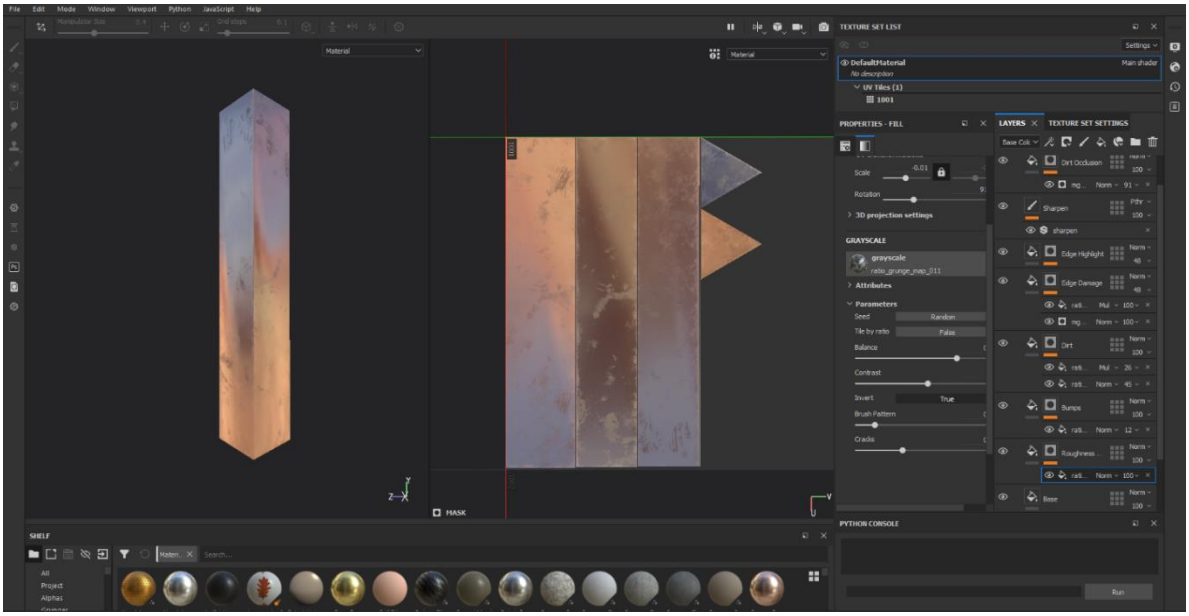
Při vytváření scény Utah Monolith jsem použil různé modely terénu vytvořené pomocí fotogrammetrie z knihovny Quixel Megascans.[53] Modely mají rozlišení textur v 4K nebo 8K v závislosti na velikosti objektu ve scéně. Jejich tvar a rozměry neodpovídají přesně skalám na fotografii, ale pro účely srovnání považuji tyto rozdíly za nedůležité. Důležitější než samotný tvar objektů, respektive jejich detaily, byl jejich povrch, textura, barva a celkový obrys a proporce. Stejně důležité bylo následné rozmístění objektů v prostoru, aby na sebe navazovaly a svíraly mezi sebou podobný úhel jako na originální fotografii. A to zejména proto, aby se dopadající světlo odráželo na podobná místa prostoru, jako je tomu v reálném snímku. Pro vytvoření modelu monolitu jsem použil software Houdini a texturu jsem vytvořil pomocí softwaru Substance painter. Abych dosáhl co nejpodobnějších vlastností materiálu monolitu, jako je odrazivost, barva nebo povrchové nedokonalosti, použil jsem jako reference různé fotografie reálného monolitu, kterých lze dohledat velké množství na internetu, a to z různých úhlů a pod různým osvětlením.

[53] KOCH, D. The Problem with the Utah Monolith. 12 30 2020.

54 55

obr. 8.1-2 Referenční snímek pro vytváření materiálu monolitu (Black 2020)

obr. 8.1-3 Referenční snímek pro vytváření materiálu monolitu (Utah monolith 2022)

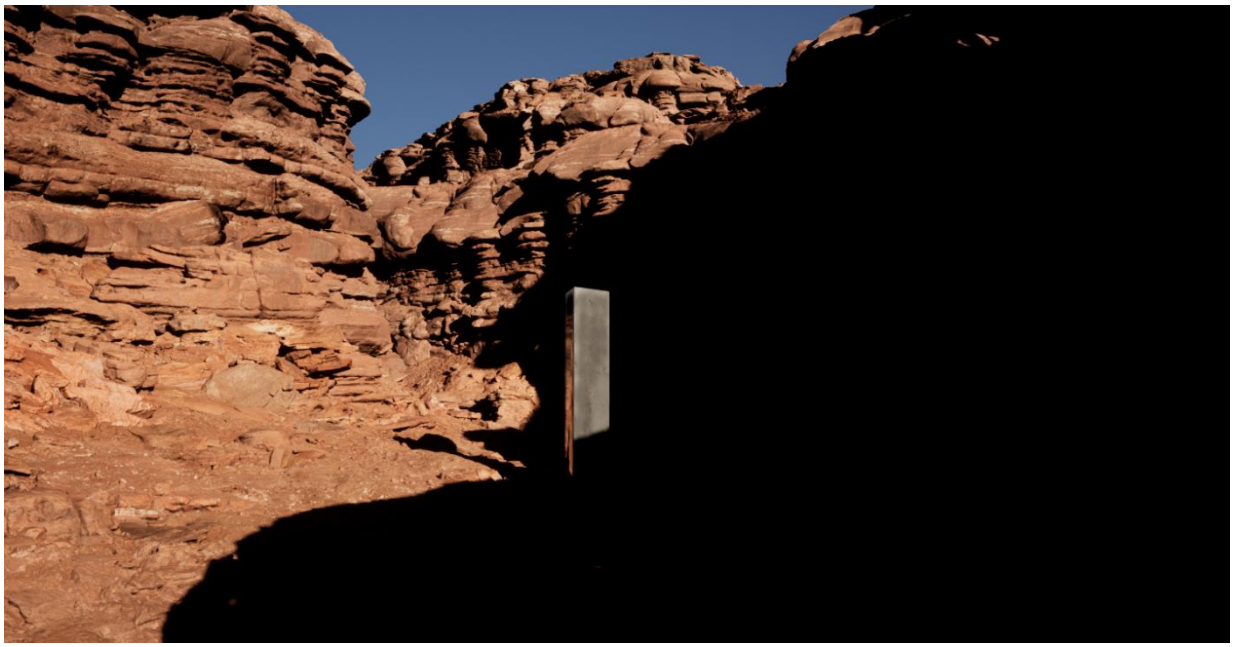


obr. 8.1-1 Vytváření povrchu monolitu v softwaru Substance painter 6.2.0, (vlastní tvorba).

[54] BLACK, R. The Magazine Of The Sierra Club. 2020.
[55] WIKIPEDIA, Utah monolith. 2022.

8.1.1 Dynamická světla

Při využití dynamických světel je důležité určit, jakým způsobem se bude realizovat globální osvětlení. Bez simulace globálního osvětlení by objekty, či jejich části, které nejsou osvětleny přímím světlem, byly zcela černé viz obr. obr. 8.1-4.



obr. 8.1-4 Zasvícená scéna s použitím dynamických světel bez simulace globálního osvětlení. Unreal Engine 4.25.4, (vlastní tvorba)

V softwaru Unreal Engine se pro aproximaci sekundárních zdrojů, zejména při práci s dynamickým osvětlením, používá tzv. Skylight. Skylight vytvoří světelný zdroj, který svítí na všechny objekty ve scéně v intenzitě udávané podle tzv. cubemap. Cubemap je „fotografie" zachycující celou scénu okolo určeného místa. Více viz. kapitola 3.5. Díky Skylight dokážeme napodobit například světlo rozptýlené v atmosféře, světlejší část oblohy odkud přichází přímé sluneční světlo, nebo odrazy světla od přímo nasvícených objektů ve scéně. Problémem je, že „fotografie", která toto světlo určuje je pro celou scénu pouze jedna. Může tak dojít k tomu, že sekundární osvětlení nebude působit stejně přirozeně pro každé místo ve scéně v závislosti na tom, odkud byla cubemapa pořízena a v jak moc rozdílném prostřední je aplikována.

Ve scéně zachycené na obrázku obr. 8.1-5 bylo použita simulace globálního osvětlení pomocí Skylight, jehož cubemap byla zachycena z bezprostřední blízkosti pozice virtuální kamery, tedy blízko středu rokle. Protože levá strana skalní rokle je osvětlena přímím světlem, aplikuje se toto světlo pomocí Skylight na pravou stranu, která byla ve stínu. Světlo rozptýlené v atmosféře a přicházející shora je v tomto případě méně intenzivní než to odražené od stěny. Již zmíněnou nevýhodou tohoto způsobu aproximace sekundárního zdroje můžeme pozorovat na nejvzdálenější části skal a na jejich vrcholu. Světlo, které na ně dopadá z levé strany není ve scéně prostorově obhajitelné žádným odrazem, ani jiným zdrojem. Přesto je aplikováno, protože pro celou scénu může být použita pouze jedna cubemap.

obr. 8.1-5 Zasvícená scéna s použitím dynamických světel se simulací globálního osvětlení pomocí Skylight. Unreal Engine 4.25.4, (vlastní tvorba)

Pokud bychom cubemap vytvořili z jiného místa na mapě, kde je světlo odráženo od svého okolí výrazně jiným způsobem, nebo které dokonce obsahuje výrazně jiné barvy (například zelenou trávu), může být simulace sekundárního osvětlení tímto ovlivněna. Výsledkem pak bude, že v místě, ze kterého je cubemap vytvořena, je sekundární osvětlení poměrně realistické, ale v jiné části naprosto neodpovídající danému prostředí.

Na obrázku obr. 8.1-6 vidíme situaci, kdy byla cubemap vytvořena z místa, kde stěny kaňonu nejsou osvětlené žádným přímím světlem. Kvůli tomu jediné sekundární osvětlení přichází ze směru oblohy a je barevně neutrální. V tmavé části kaňonu nedochází k dopadu světla odraženého od protilehlé stěny, a tak scéna působí méně realisticky.

obr. 8.1-6 Zasvícená scéna s použitím dynamických světel se simulací globálního osvětlení pomocí Skylight. Cubemap vytvořená v místě, kde přímé světlo neosvětluje žádné okolní svěny kaňonu. Unreal Engine 4.25.4, (vlastní tvorba)

Pokud by mapa, kterou se snažíme zasvítit obsahovala více různě barevných prostředí, vedlo by vytvoření cubemap z jednoho prostředí přenos odražených barev do prostředí druhého. Na obr. 8.1-7 vidíme situaci, kde červené skály kaňonu přecházejí do zelenějšího prostředí. Cubemapa byla vytvořena z prostředka tohoto zelenějšího prostředí. Cubemapa však ovlivňuje i prostředí skalnatého kaňonu s monolitem viz obr. 8.1-8. Do stínu na pravé straně obrazu se jasně propisují barvy z prostředí s trávou.

obr. 8.1-7 Pohled z místa porostlého trávou, odkud byla vytvořena cubemap. Unreal Engine 4.25.4, (vlastní tvorba)

obr. 8.1-8 Zasvícená scéna s použitím dynamických světel se simulací globálního osvětlení pomocí Skylight. Cubemap vytvořená v místě porostého trávou. Zelené tóny sekundárních zdrojů osvětlení se propisují do zastíněných částí obrazu. Unreal Engine 4.25.4, (vlastní tvorba)

Problém různě barevných prostředí, a tedy ovlivnění barev sekundárního osvětlení je možné řešit tak, že zcela odfiltrujeme barvy prostředí kromě barvy oblohy, u které se dá předpokládat, že bude ve všech místech scény stejná. Můžeme využít oblohu, kterou ve scéně simulujeme, nebo nahrát obrázek oblohy vlastní viz. obr. 8.1-10. Tím docílíme konstantnějšího dojmu v celé mapě. Nevýhodou je, že tím také potlačíme simulaci odrazů od okolních objektů v mapě viz obr. 8.1-9.

obr. 8.1-9 Zasvícená scéna s použitím dynamických světel se simulací globálního osvětlení pomocí Skylight. Použita je vlastní cubemap viz obr obr. 8.1-10. Unreal Engine 4.25.4, (vlastní tvorba)

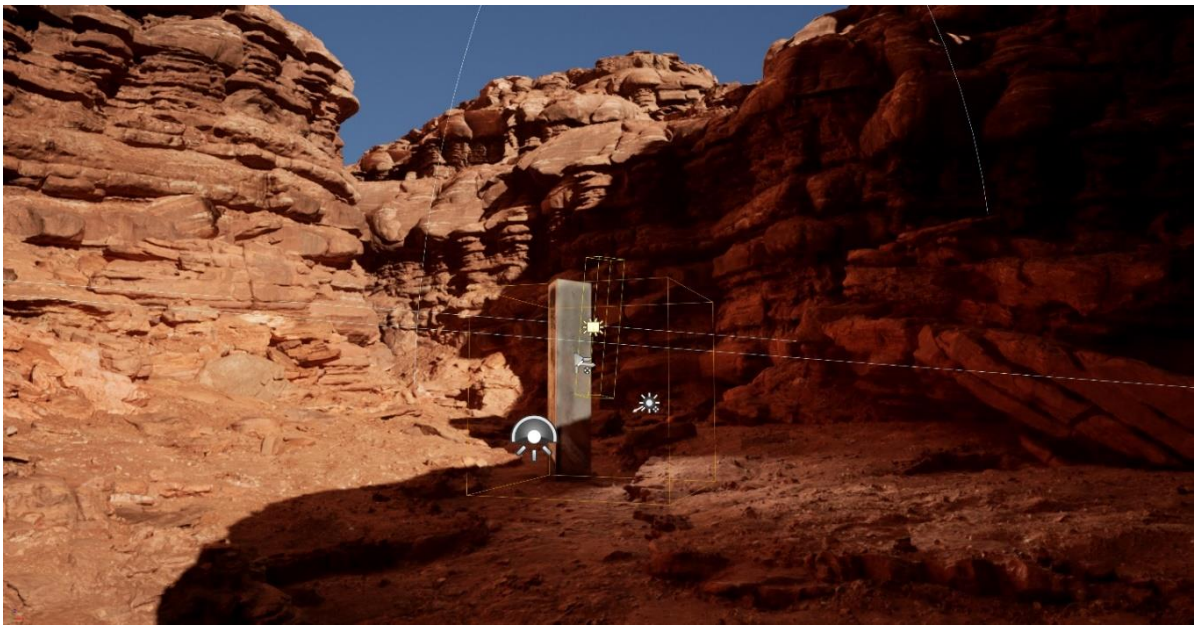
56

obr. 8.1-10 cubemap použitá ve scéně zobrazené na obr. 8.1-9. (Dimitrios Savva nedatováno)

[56] SAVVA, D. ZAAL G. Poly Haven.

Silný odraz slunečního světla od lesklého kovového monolitu, který vidíme na originální fotografii, není možné simulovat přímo ze směrového světla, které reprezentuje sluneční svit. Pokud chceme docílit tohoto efektu, je potřeba využít další světelný zdroj. Já se rozhodl použít plošné světlo, které se ovšem v případě nastavení mobility na dynamické chová velmi podobně jako bodové světlo, které by bylo také možné použít. Nevýhodou tohoto postupu je, že při změně úhlu dopadu „slunečního“ světla se nezmění směr, pod kterým je světlo simulující odraz vyzařováno. To by bylo možné řešit naprogramováním pohybu světla v závislosti na pohybu slunce po obloze. Pro účely statického obrazu to však není potřeba.

obr. 8.1-11 Zasvícená scéna s použitím dynamických světel se simulací globálního osvětlení pomocí Skylight a doplněná o plošné světlo simulující odraz od kovového monolitu. Unreal Engine 4.25.4, (vlastní tvorba)



obr. 8.1-12 Umístění plošného zdroje světla simulujícího odraz slunečního svitu od kovového monolitu. Unreal Engine 4.25.4, (vlastní tvorba)

Obnovovací frekvence snímků v této scéně dosahovala přibližně 45fps a to při použití pouhých třech světelných zdrojů. Je potřeba brát v potaz, že tento údaj je ovlivněn měřením obnovovací frekvence při scéně otevřené v editoru. Při spuštění samostatné aplikace, ne editoru, by byla obnovovací frekvence vyšší. Pro srovnání je tento údaj však dostatečný.

8.1.2 Statická světla

Použití statických světel vyžaduje značný objem práce navíc ve srovnání s použitím dynamických světel. Je nutné vytvořit precizní UV mapy ze kterých se následně generují UV lightmapy. Bez nich by statická světla nešlo vůbec použít. Pro tvůrce je také s tímto typem světel výrazně méně pohodlné pracovat. Po každé změně pozice či jiného nastavení světel, je nutné pro získání výsledného zobrazení vyrenderovat všechny lightmapy znovu. Což může v některých případech trvat i několik hodin.

I když by statická světla měla dosahovat věrnější simulace osvětlení, a tedy lepších vizuálních výsledků, dynamické svícení je tak pokročilé, že rozdíl může být v některých případech jen minimální. Hlavní benefit statického svícení je však v mnohonásobném zvýšení obnovovací frekvence. Ta vzrostla ze zmíněných 45fps při použití dynamických světel na maximálních 120fps při použití statických světel.

obr. 8.1-13 Zasvícená scéna s použitím statických světel. Unreal Engine 4.25.4, (vlastní tvorba)

Na první pohled se může zdát, že výsledky statického a dynamického svícení jsou velmi podobné, ale při bližším pohledu je vidět, že sekundární osvětlení je preciznější v distribuci světelných odrazů z osvětlených míst na neosvětlená viz obr. 8.1-14.

obr. 8.1-14 Porovnání scény vygenerované pomocí statických světel (vlevo) a scény vygenerované pomocí dynamických světel (vpravo). Unreal Engine 4.25.4, (vlastní tvorba)

9 | Scéna 3 – Interiér bytu – den

Ve třetí scéně jsem nevycházel z reálné fotografie, ale použil existující komponenty z ukázkové scény od výrobce softwaru Unreal Engine[57] které jsem upravil a rozmístil tak, aby na nich mohly být demonstrovány různé druhy svícení. Důvod, proč jsem nevycházel z reálné fotografie je zejména časová náročnost výroby potřebných modelů a to, že na této scéně chci zejména testovat, jakých výsledků dosahují jednotlivé přístupy a jaký má každý jednotlivý světelný zdroj a komponent ovlivňující simulaci osvětlení vliv na výsledný obraz. Proto není srovnání s reálnou fotografií tolik podstatným.

Jako výchozí světelnou atmosféru jsem vybral den, kdy do místnosti pronikají jak přímé sluneční paprsky, tak i velké množství odraženého světla od protější budovy i světla rozptýleného v atmosféře. Tuto kombinaci doplňuje interiérové osvětlení. Vybral jsem ji proto, že se jedná o velmi častou světelnou situaci, a tak ji máme dobře uloženou v paměti, jako určitý referenční obraz. A také proto, že ať už v praxi kameramana hraného filmu, nebo umělce, který zasvětluje virtuální scény se taková atmosféra bude vyskytovat velmi často. Oproti předchozím scénám je tu více různých zdrojů světla, což sice ztěžuje orientaci, jakým způsobem se jednotlivé zdroje chovají, ale na druhou stranu dobře ukazuje, jak se vzájemně ovlivňují a jak s nimi můžeme pracovat.
[58]

obr. 8.1-1 Nenasvícené scéna interiéru s ikonami umístění jednotlivých světel. Unreal Engine 4.25.4, (vlastní tvorba, (Engine 2019))

[57] ENGINE, U. New Archviz Interior Rendering sample. 2019.
[58] ENGINE, U. New Archviz Interior Rendering sample. 2019.

9.1 Porovnání dosažených výsledků různých metod renderování scény

V tomto porovnání se již nebudu zabývat tím, jaké jsou výhody a nevýhody jednotlivých typů zasvětlení scény. Jde čistě o vizuální porovnání dosažených výsledků. Všechny světelné zdroje byly shodné intenzitou, umístěním, barevnou teplotou i ostatním nastavením. Jediný rozdíl v nastavení světelných zdrojů je nastavení mobility a v případě světelného zdroje Skylight nastavením vzdálenosti ze které vytváří cubemap. O tomto rozdílu nastavení píši dále v textu. Je důležité zdůraznit, že výsledek zasvícení scény nezávisí pouze na nastavení světelných zdrojů, ale i na mnoha dalších aspektech. Zejména na kvalitě UV map, textur, nastavení, jak mají objekty na světlo reagovat, nebo nastavení renderu. Zejména při nastavení renderování lightmap při použití statického osvětlení může docházet k velmi významným rozdílům ve výsledné podobně obrazu. Proto je toto porovnání potřeba brát s určitou rezervou a má sloužit jako ukázka, jaké výsledky můžeme očekávat.

obr. 9.1-1 Nastavení renderu lightmap pro statické osvětlení. Unreal Engine 4.25.4, (vlastní tvorba)

Rozmístění světel ve scéně, viz obr. obr. 8.1-1, je poměrně přímočaré. Světla umělá jsou v místech svých přirozených zdrojů. Jedná se o reflektorová světla (spotlight) v kombinaci se světlo emitující žárovkou, která ovšem nemá na své okolí vliv ve smyslu osvícení předmětů. Světlo směrové (directional light) nijak neovlivňuje jeho umístění, protože paprsky z něj vycházející jsou vždy rovnoběžné a dopadající na všechny celou scénu. Umístění tedy může být libovolné a jediné co nastavujeme je úhel pod kterým paprsky dopadají. Posledním typem použitého světelného zdroje je Skylight. V případě skylight záleží jednak na umístění zdroje, ale také na nastavení, z jaké minimální vzdálenosti má vytvářet cubemap. Protože při renderování globálního osvětlení pomocí ray-tracingu a statického osvětlení je přirozenější simulovat atmosférou rozptýlené světlo přímo z okolní atmosféry, je u těchto dvou typech svícení skylight nastaven tak, aby cubemap nevytvářel z interiéru, ale právě z okolí domu. To však není vhodné v případě dynamického osvětlení, protože v tomto případě není světlo vytvořené ze Skylight modulováno okny bytu a je na všechny předměty aplikováno, jako kdyby se nacházely venku. Proto je Skylight nastaven tak, aby cubemap vytvářel ze světla dopadajícího dovnitř bytu, tedy slunečních paprsků a umělého osvětlení. Díky tomu je možné simulovat mnohem přesněji sekundární osvětlení uvnitř místnosti. Pokud by se jednalo například o počítačovou hru, ve které hráč prochází z jedné místnosti do druhé, nebo z exteriéru do interiéru, je možné vytvořit pro každý prostor samostatný Skylight a omezit jejich působnost pouze na daný prostor.

obr. 9.1-2 Porovnání scény zasvícené pomocí dynamického osvětlení s různým nastavením zdroje Skylight. Zdroj Skylight vytváří cubemap z prostoru pokoje (vlevo). Zdroj Skylight vytváří cubemap z atmosféry v okolí domu (vpravo). Unreal Engine 4.25.4, (vlastní tvorba, (Engine 2019))

Při porovnání obrazů scén získaných jednotlivými přístupy jako celku, první aspekt a asi nejvýraznější rozdíl mezi nimi je různé barevné podání. Zatímco mezi ray-tracingem a Statickým osvětlením není rozdíl příliš velký, u dynamického osvětlení je odlišnost značná. Hlavním důvodem pro tento rozdíl je světelný zdroj Skylight. Protože značný podíl na osvětlení scény má sekundární osvětlení, jehož původce je v případě dynamického osvětlení právě Skylight, je velmi podstatné jeho umístění a nastavení. Pokud bychom například posunuli Skylight blíže ke stolu, na který svítí interiérové osvětlení, větší plochu vytvořené cubemap by zabíraly teplé tóny z těchto světel vycházející. Tím by došlo k emisi tohoto teplého světla na okolní předměty a celá scéna by se jevila teplejší. Naopak posunutím Skylight k oknu by větší dominanci převzalo sluneční světlo, které má vyšší teplotu chromatičnosti, a tak by se místnost jevila chladnější.

obr. 9.1-3 Scéna zasvětlená pomocí ray-tracingu. 20fps. Unreal Engine 4.25.4, (vlastní tvorba), (Engine 2019)

obr. 9.1-4 Scéna zasvětlená pomocí dynamického osvětlení. 43fps. Unreal Engine 4.25.4, (vlastní tvorba), (Engine 2019)

obr. 9.1-5 Scéna zasvětlená pomocí statického osvětlení. 120fps. Unreal Engine 4.25.4, (vlastní tvorba), (Engine 2019)

Jedním z hlavních aspektů, které vytváří dojem prostorovosti, a tedy i realističnosti obrazu jsou stíny. Na obr. 9.1-6, vidíme, že zejména jemné stíny, které vrhají objekty na své okolí se svou kvalitou značně liší podle použité techniky. Kupříkladu při použití Statického osvětlení je jasně viditelný stín pod papírem, který leží na stole v pravé spodní části obrazu. Na obrázku ve středu, který je vytvořený pomocí ray-tracingu vidíme, že oproti dynamickému i statickému osvětlení je vykreslený stín stojací lampy na pravé stěně, nebo stín větráku na stropě. Jestli je zobrazení s těmito stíny, či bez nich věrnější realitě je obtížné posoudit.

Ve stínování při použití ray-tracingu si však můžeme všimnout poměrně viditelného šumu a nerovnoměrností. Například stíny pod oknem v okolí topeného tělesa jsou na některých místech tmavší než na jiných. To v případě statického osvětlení jsou stíny velmi konzistentní. U Dynamického osvětlení stíny objektů od sekundárního osvětlení nejsou téměř žádné. Je to z toho důvodu že světlo zajišťující globální osvětlení u dynamického osvětlení Skylight tyto stíny vytvářet nedokáže. Jediný komponent osvětlení, který alespoň částečně tyto stíny vytváří je Ambient occlusion a Screen Space Ambient Occlusion, které však nedosahují takových výsledků jako ray-tracing a statické osvětlení.

Pokud bychom například chtěli snížit výpočetní náročnost při výpočtu globálního osvětlení, ale zároveň získat stíny, jako při použitá ray-tracingu, můžeme ray-tracing použít například pouze na stíny, případně na jiné efekty jako například renderování průhledných objektů, výpočet refrakce skla, nebo odrazy v lesklých materiálech.

obr. 9.1-6 Srovnání výřezů obrazu třech způsobů zasvícení scény. Statické osvětlení (vlevo), ray-tracing (uprostře), dynamiské osvětlení (vpravo). Unreal Engine 4.25.4, (vlastní tvorba, (Engine 2019))

9.2 Dynamický objekt v různých typech nasvícení

Scéna, kterou jsme do této chvíle porovnávali byla celá složena pouze z objektů, se kterými nelze pohybovat. Pokud do scény ale vložíme objekt, který je určen k tomu, aby se pohyboval, jako například postavu, tak v případě ray-tracingu a dynamického osvětlení se nic nemění. Obě tyto zasvícení budou fungovat stejně jako se statickými objekty. V případě statického osvětlení však musíme vytvářet lightmapy a to nelze pro dynamické objekty. Proto se v případech, kdy potřebujeme dynamické objekty, vytváří tzv. Volumentric Lightmaps viz. kapitola 2.2. V místech, kde má světlo interagovat s těmito objekty používáme místo statických světel světla stacionární. Volumetric lightmap simuluje sekundární osvětlení na dynamických objektech pomocí předem vypočítaného osvětlení v rozmístěných bodech v prostoru. Nastavení počtu a rozmístění těchto bodů je zásadní pro kvalitu efektu sekundárního osvětlení. Počet bodů a množství informace v nich uložené ale také velmi ovlivňuje velikost scény v operační paměti a zvyšuje nároky na grafickou kartu. Pomocí Volumetric lightmaps můžeme simulovat sekundární osvětlení, ale ne přímé, tedy primární osvětlení. Primární osvětlení statických světel nemá žádný vliv na dynamické objekty, protože jejich efekt je vepsán pouze do lightmap. Pokud chceme, aby přímé osvětlení ovlivňovalo i dynamické objekty, musíme využít tzv stacionární osvětlení viz kapitola 4.1.3. Stacionární osvětlení kombinuje funkce statického světla, to znamená generování lightmap i volumetric lightpam, ale zároveň se chová jako dynamické světlo s tím rozdílem, že s ním nemůžeme pohybovat. Jinak řečeno, funguje podobně, jako bychom na místo statického světla umístili i světlo dynamické se stejným nastavením. Obrázek obr. 9.2-1 ukazuje použité rozmístění bodů Volumetric lightmap.

obr. 9.2-1 Rozmístění bodů Volumetric lightmap. Unreal Engine 4.25.4, (vlastní tvorba, (Engine 2019))

Ve srovnání nasvícení dynamického objektu, viz obr. 9.2-2, vidíme, že při použití statického osvětlení je sice sekundární osvětlení nejkvalitnější, ale na postavu nedopadá přímé světlo z lampy nad stolem. V případě ray-tracingu a dynamického osvětlení je přímé osvětlení nad stolem sice jasně vykresleno na postavě, ale sekundární osvětlení je méně výrazné.

obr. 9.2-2 Srovnání nasvícení dynamického objektu ve třech způsobech zasvícení scény. Statické osvětlení (vlevo), ray-tracing (uprostřed), dynamické osvětlení (vpravo). Unreal Engine 4.25.4, (vlastní tvorba, (Engine 2019))

10 | Zhodnocení výsledku experimentu

Srovnáme-li všechny přístupy simulace osvětlení všech scén, můžeme konstatovat, že obecně Statické osvětlení dosahuje nejlepších vizuálních výsledků. Zároveň však přináší nejvíce omezení v podobě omezené funkcionality pro dynamické objekty, nemožnosti používat pohyblivá světla a v neposlední řadě výrazně komplikovanější výrobě 3D modelů i postupu při zesvětlování scény. Zároveň je ale potřeba zdůraznit, že lepších vizuálních výsledků nemusí statické osvětlení dosahovat ve všech případech. Jak ukazuje první scéna - 2001: A Space Odyssey, kde byla vizuální podoba všech přístupů velmi podobná.

Pokud porovnáme obnovovací frekvence, je nesporné, že statická světla jsou nejvhodnějším řešením, protože nestojí žádný výpočetní výkon při běhu programu. Jejich použitelnost je ale v praxi velmi omezena kvůli tomu, že neinteragují s pohyblivými objekty. Pro situace, kde je vizuální dojem prioritou, kde pohyb světel ani objektů není nutný a kde máme k dispozici kvalitní UV mapy, jsou Statická světla nejlepším řešením. To v praxi mohou být například architektonické vizualizace.

Dynamické osvětlení, se ve výpočetní náročnosti řadí, v porovnání těchto tří přístupů, doprostřed. Světla se sice mohou pohybovat a reagují se všemi objekty ve scéně, ale nevytváří žádné, nebo jen velmi přibližné sekundární osvětlení v závislosti na použité technologii. Kvůli tomu dosahuje dynamické osvětlení z porovnávaných třech nejhorších vizuálních výsledků. Použití dynamického osvětlení je ale vhodné zejména pro počítačové hry, kde se předpokládá interakce s objekty scény, případně se mění nasvícení scény v průběhu hraní.

Ray-tracing do určité míry spojuje výhody dvou předchozích přístupů. Vytváří dynamické sekundární osvětlení, které sice není tak kvalitní, jako v případě statického osvětlení, ale je výrazně přesnější než v případě dynamického osvětlení. Se světly při použití ray-tracingu můžeme libovolně pohybovat a měnit jejich nastavení při chodu programu. Jedná se tak o nejuniverzálnější nástroj při zesvětlování scény, ovšem za cenu nejvyšších výpočetních nároků.

Při práci na komplexnějším projektu je běžné, že se jednotlivé přístupy kombinují pro dosažení optimální vizuální kvality, při zachování požadované interaktivity a za přijatelných výpočetních nároků. Ray-tracing je možné použít pouze na určité efekty jako odlesky, stíny či jen na některé světelné zdroje. Statické osvětlení zle kombinovat s dynamickým v podobě stacionárních světel a globální osvětlení je i v případě dynamického osvětlení možné simulovat různými postupy v závislosti na technologiích dostupných v používaném softwaru.

Cílem tohoto srovnání je dát základní představu o aktuálních možnostech, které zmiňované postupy nabízejí a jimi dosažitelných výsledcích, ne o návod, jak dosáhnout ideálních výsledků jak ve vizuálním aspektu obrazu, tak v optimalizaci výpočetních nároků. To zejména proto, že ke každé scéně a každému projektu je nutné přistupovat individuálně podle jeho určení a obsahu.

11 | Závěr

Vytváření obrazu pomocí editorů pro real-time aplikace nabízí mnoho nových možností a může dosahovat vizuálních kvalit srovnatelných s postupy klasického renderování, ovšem za cenu poměrně značných omezení. Tvůrce musí dobře znát cíl, kterého se snaží dosáhnout, aby mohl použít pouze takové techniky, které jsou pro požadovaný výsledek nezbytné a těm, které by dílu škodily se vyhnout. Zároveň by měl vědět, jak se jednotlivé technologie chovají při použití v různých typech scén a při různých způsobech nasvícení a jaké nároky na výpočetní výkon mají. Mnohdy tak může předejít zbytečnému zatěžování počítače v případech, kdy bylo možné použít výrazně úspornější metodu s minimálním vizuálním rozdílem, nebo dokonce s výrazně lepším vizuálním výsledkem.

Hlavní oblastí, kterou jsem zkoumal v praktické části práce byly tři technologicky rozdílné přístupy k simulaci osvětlení ve virtuální scéně. Ray-tracing, statické osvětlení, které ukládá informaci o osvětlení do lightmap, a dynamické osvětlení, které využívá výhradně rasterizaci bez použití lightmap. Jako nejuniverzálnější a zároveň nejpohodlnější k používání se jeví ray-tracing. Pokud používáme ray-tracing, není nutné vytvářet lightmapy, které často trvají velmi dlouho vygenerovat. Také není nutné mít UV lightmapy použitých modelů. Omezením, se kterým musíme při používání ray-tracingu počítat je jeho vysoká výpočetní náročnost. Pokud bychom se měli zaměřit pouze na kvalitu distribuce světla ve scéně, s největší přesností a díky tomu často i s nejlepším vizuálním výsledkem funguje statické osvětlení. Ovšem za cenu největších omezení spojených s interaktivitou. Rozumným kompromisem mezi těmito metodami může být dynamické osvětlení, které při nízké hardwarové náročnosti může dosahovat dostatečných vizuálních výsledků a s plnou interaktivitou mezi objekty, světlem a efekty ve scéně. Jak se ukázalo, každý z těchto přístupů má své přednosti a zápory a nedá se říct, že by se některý z nich hodil pro všechny aplikace.

Jak v teoretické, tak praktické části mé diplomové práce jsem popsal mnoho technologií, které se používají k simulaci různých světelných jevů. Real-time aplikace jsou v tomto ohledu velmi specifické. Při dnešních výpočetních výkonech běžných počítačů není možné použít nějaký zcela univerzální nástroj pro simulaci osvětlení v komplexní scéně. Jednotlivé technologie, nástroje a metody se kombinují, a tak se snadno celý proces může stát nepřehledným a matoucím. Jedním z mých cílů bylo podat co možná nejkomplexnější pohled na tuto problematiku, ale zároveň tak, aby se stala snadno přehlednou zejména pro tvůrce bez předchozí hlubší znalosti problematiky real-time CGI nebo CGI obecně. Pokud tvůrce ovládne všechny tyto techniky, lze vytvářet scény ve velmi podobné kvalitě, jako je obraz generovaný pomocí offline renderovacích technik, může tvořit svobodně jako při svícení scén v reálném světě a může využívat výhod, které mu simulace osvětlení v real-time CGI aplikacích nabízí. Vytvářet realitu, která není pouze odrazem našeho světa, ale která je realitou zcela novou.

12 | Seznam použitých pramenů a literatury

12.1 Online zdroje

BLACK, R. The Magazine Of The Sierra Club. 19 12 2020.
https://www.sierraclub.org/sierra/utah-monolith-just-latest-tale-desert-trash (accessed 6 14, 2022).

CADKID69. Baked lightmap from blender. 2020.
https://i.redd.it/ho75572lpw931.png (accessed 6 21, 2021).

CAULFIELD, B. Nvidia Blogs. 19 3 2018.
https://blogs.nvidia.com/blog/2018/03/19/whats-difference-between-ray-tracing-rasterization/ (accessed 5 18, 2021).

CABELEIRA, J. Combining Rasterization and Ray Tracing Techniques to Approximate Global Illumination in Real-Time. Técnico Lisboa, 2010.

COMPANY, SUDO NULL. Reflective Shadow Maps: Part 1. 2019.
https://sudonull.com/post/100-Reflective-Shadow-Maps-Part-1 (accessed 7 30, 2021).

CYCU1. Cyberpunk 2077 Ray Tracing ON vs OFF RTX 3080 4K Graphics Comparison. 18 12 2020.
https://www.youtube.com/watch?v=DSst2XFtMTU (accessed 6 2, 2021).

KOCH, D. The Problem with the Utah Monolith. 12 30 2020.
https://petapixel.com/2020/11/30/the-problem-with-the-utah-monolith/ (accessed 6 1, 2022).

CAMBRIDGE DICTIONARY. Meaning of shadow in English. n.d.
https://dictionary.cambridge.org/dictionary/english/shadow (accessed 07 23, 2021).

SAVVA, D. ZAAL G. Poly Haven. n.d.
https://polyhaven.com/a/belfast_sunset (accessed 06 27, 2022).

DOCUMENTATION, Unreal Engine 4.26. Rect Lights. Epic Games, Inc. n.d.
https://docs.unrealengine.com/4.26/enUS/BuildingWorlds/LightingAndShadows/LightTypes/RectLights/ (accessed 7 6, 2021).

UNREAL ENGINE. New Archviz Interior Rendering sample. 19 12 2019.
https://www.unrealengine.com/en-US/blog/new-archviz-interior-rendering-sample-project-now-available?sessionIsunnvalidated=true (accessed 6 25, 2022).

EVGA. Flee The Shattered Ruins Of Moscow. 2022.
https://www.evga.com/articles/archive/01309/metro-exodus/default.asp (accessed 1 16, 2022).

EPIC GAMES, UE4 Sun Temple. 23 3 2014.
https://www.unrealengine.com/marketplace/en-US/learn/sun-emple/reviews?sessionInvalidated=true.

HAINES E. Ray Tracing Essentials, Part 1: Basics of Ray Tracing. Produced by NVIDIA. 2019.
Hodilton. Control - RTX ON vs RTX OFF Comparison - Real Time Ray Tracing - Max Settings - 4k. 12 11 2019.
https://www.youtube.com/watch?v=45pfvYX-fxU&t=1s (accessed 6 2, 2021).

KESHAV, CH. Light mapping - Theory and amplementation. 21 7 2003.
https://www.flipcode.com/archives/Light_Mapping_Theory_and_Implementation.shtml (accessed 6 20, 2021).

ICYOU520. Eevee vs Cycles Comparisons. 1 březen 2018.
https://blenderartists.org/t/eevee-vs-cycles-comparisons/1101050 (accessed 6 1, 2021).

KASYAN, N. Playing with Real-Time Shadows. 2013.
https://docplayer.net/32214142-Playing-with-real-time-shadows-nikolas-kasyan-crytek.html (accessed 7 30, 2021).

HEARN, P. What is Path Tracing and Ray Tracing? And Why do They Improve Graphics?, 2019, vol. 2021.
https://www.online-tech-tips.com/computer-tips/what-is-path-tracing-and-ray-tracing-and-why-do-they-improve-graphics/d

LAMPEL, J. Cycles vs. Eevee - 15 Limitations of Real Time Rendering in Blender 2.8. 7 březen 2019.
https://cgcookie.com/articles/blender-cycles-vs-eevee-15-limitations-of-real-time-rendering (accessed 6 1, 2021).

LEACH, C. failengine. 2019.
http://www.craigleach.nl/failengine/ (accessed 7 30, 2021).

EPIC GAMES. mobility, Light. Movable Lights. Epic Games. n.d.
https://docs.unrealengine.com/4.27/enUS/BuildingWorlds/LightingAndShadows/LightMobility/DynamicLights/ (accessed 06 10, 2022).

MUNDRA, ANISH. WHAT Is Ray Tracing? 21 9 2021.
https://www.studytonight.com/post/what-is-ray-tracing (accessed 1 15, 2022).

EPIC GAMES. Rect Lights. Epic Games, Inc. n.d.
https://docs.unrealengine.com/4.26/enUS/BuildingWorlds/LightingAndShadows/LightTypes/RectLights/ (accessed 7 6, 2021).

REED, N. Mirror Reflections,
https://computergraphics.stackexchange.com/questions/4585/mirror-reflections-ray-tracing-or-rasterisation. (accessed 07 16, 2021).

SOLOMON, J. "Introduction to Computer Graphics (Lecture 16): Global illumination." 2020.
https://www.youtube.com/watch?v=odXCvJTNn6s&t=1769s

TYSON, M. DirectX Raytracing tech demo video published by Futuremark. 21 březen 2018.
https://hexus.net/tech/news/graphics/116438-directx-raytracing-tech-demo-video-published-futuremark/ (accessed 6 2, 2021).

WIKIPEDIA. Utah monolith. 7 6 2022.
https://cs.wikipedia.org/wiki/Kovov%C3%BD_monolit_v_Utahu (accessed 6 14, 2022).

VRIES, J. Basic Lighting.
https://learnopengl.com/Lighting/Basic-Lighting (accessed 7 5, 2021).

WESTER, A. Ray-tracing soft shadows in real-time. 20 1 2020.
https://medium.com/@alexander.wester/ray-tracing-soft-shadows-in-real-time-a53b836d123b
(accessed 7 12, 2021).

WOLFE, A. Programming, Graphics, Gamedev, Exotic Computation, Audio Synthesis. 2019.
https://blog.demofox.org/2016/09/21/path-tracing-getting-started-with-diffuse-and-emissive/ (accessed 2 16, 2021).

12.2 Tištěné zdroje

BAILEY, M. Graphics Shaders Theory and Practice. Edtion ed. New York: CRC Press, 2012. ISBN: 978-1-4398-6775-4.

RAMOS, B. B., DORAN, J. P. Unreal Engine 4 Shaders and Effects Cookbook: Over 70 recipes for
mastering post-processing effects and advanced shading techniques. Packt Publishing, 2019. ISBN: 1789538548.

KLAWONN, F. Introduction to Computer Graphics . London: Springer, 2008. ISBN: 978-1-84628-847-0.

GLASSNER, A. S. *An Introduction to Ray Tracing*. Edtion ed. San Diego: Academic Press INC., 1989.
327 p. ISBN 9780122861604.

BOKSANSKY J., WIMMER M., BITTNER J. "Ray Traced Shadows: Maintaining Real-Time Frame
Rates." Ray Tracing Gems. Prague: Czech Technical University in Prague, 2019.

FOLEY J. D., VAN DAM A., FEINER S K., HUGHES, J.
Computer Graphics: Principles and Practice. United States: Addison-Wesley, 1995. ISBN: 978-0321399526.

KURACHI, N. The Magic of Computer Graphics. Tokyo, Japan: Ohmsha, Ltd.,, 2007. ISBN: 978-1-56881-577-0.

HUMES, L. E., BUSEY, T. A., CRAIG, J. C., KEWLEY-PORT, D.
Atten Percept Psychophys. Bloomington, Indiana: Indiana University, 2009. doi: 10.3758/APP.71.4.860

MCREYNOLDS, T. Advanced Graphics Programming Using OpenGL. Elsevier Inc., 2005. ISBN-13: 978-1558606593

SLUSALLEK P., GEORGIEV, I., ENGELHARDT, T., PHILLIPS, M., DAVIDOVIC, T., DACHSBACHER, C., "3D Rasterization–Unifying Rasterization and Ray Casting." (Universität des Saarlandes) 2009.

ROHÁČEK, D. Improving probes in dynamic diffuse global illumination
Brno: Vysoké Učení Technické v Brně, 2018.

PARK S., BAEK, S. "A Shader-Based Ray Tracing Engine." Applied Sciences, 2021, 11 ed.

TUJULA, A. Lighting and normal mapping in computer Graphics. Turku University of Applied Sciences, 2016.

VRIES, J. Learn OpenGL: Learn modern OpenGL graphics programming in a step-by-step fashion.
Kendall & Welling, 2020. ISBN: 9090332561.

WHITE S, COULTER V. N. D., JACOBS M., SATRAN M,. Cascaded Shadow Maps. In.: Microsoft,
2020, vol. 2022, p. 1.

12.3 Seznam citovaných filmů

KUBRICK, S. 2001: A Space Odyssey, Stanley Kubrick Productions, Great Britain, 1968