VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZOBRAZOVÁNÍ VIRTUÁLNÍCH 3D SCÉN S VYSOKÝM DYNAMICKÝM ROZSAHEM

HIGH DYNAMIC RANGE RENDERING OF 3D GRAPHIC SCENES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MICHAL ROZENBERG**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAN PEČIVA, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

150561

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Rozenberg Michal**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Zobrazování virtuálních 3D scén s vysokým dynamickým rozsahem**
Kategorie:	Počítačová grafika
Akademický rok:	2022/23

Zadání:

1. Nastudujte si problematiku zobrazování virtuálních 3D scén s vysokým dynamickým rozsahem (HDR) a technologie, které se k tomu používají.
2. Navrhněte aplikaci demonstrující vybrané metody zobrazování scén s vysokým dynamickým rozsahem a využívající API Vulkan.
3. Aplikaci implementujte.
4. Vyhodnoťte zkušenosti s implementovanými algoritmy a diskutujte výsledky. Porovnejte vizuální kvalitu při rendrování s vysokým dynamickým rozsahem a bez něj.
5. Práci zveřejněte na internetu. Zvažte zpřístupnění zdrojového kódu pod některou z open-source licencí.

Literatura:

- dle instrukcí vedoucího

Při obhajobě semestrální části projektu je požadováno:
Bez požadavků.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Pečiva Jan, Ing., Ph.D.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	10.2.2023

Abstrakt

Výstupem této bakalářské práce je demonstrační aplikace implementující techniky a algoritmy používané při vykreslování grafických 3D scén ve vysokém dynamickém rozsahu. K implementaci této aplikace je využito programovacího jazyka C++ a nízkoúrovňové rozhraní Vulkan API pro programování náročných 3D grafických programů.

Abstract

The output of this bachelor thesis is a demonstration application that implements techniques and algorithms used in rendering graphical 3D scenes in high dynamic range. The C++ programming language and the low-level Vulkan API are used to implement this application.

Klíčová slova

Vysoký dynamický rozsah, HDR, mapování tónů, zpracování obrazu, 3D renderování, 3D scéna, C++, Vulkan API.

Keywords

High dynamic range, HDR, tone mapping, image processing, 3D rendering, 3D scene, C++, Vulkan API.

Citace

ROZENBERG, Michal. *Zobrazování virtuálních 3D scén s vysokým dynamickým rozsahem.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Pečiva, Ph.D.

Zobrazování virtuálních 3D scén s vysokým dynamickým rozsahem

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jana Pečivy, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Rozenberg
10. května 2023

Poděkování

Chtěl bych poděkovat především panu Ing. Janu Pečivovi, Ph.D. za jeho vstřícnost, ochotu, trpělivost a za věcné odborné rady, které mi velmi pomohly při vypracovávání této bakalářské práce. Poděkování patří také S.S. a R.R.

Obsah

Kapitola 1

Úvod

Věc vysokého dynamického rozsahu je v současné době velmi aktuální. Zajímají se o ni ne jen vědecké práce, ale také především společnosti zaměřující se na výrobu a prodej elektroniky a úměrně s tím roste i zájem veřejnosti, čili spotřebitelů. V oboru fotografie je vysoký dynamický rozsah známý už mnoho let, ale do oblasti vykreslování 3D grafiky se prosadil až v pozdních začátcích nového tisíciletí a to prostřednictvím herního průmyslu. Tato práce si tak klade za cíl uvést do souvislostí veškeré pojmy, které se k vysokému dynamickému rozsahu v oblasti 3D počítačové grafiky vážou. První polovina práce je zaměřena na studium této problematiky a uvedení do kontextu, až se v závěru této poloviny upíná na konkrétní problémy z oblasti mapování tónů. Do druhé poloviny uvede návrh výstupní aplikace, která má prezentovat problematiku vysokého dynamického rozsahu v praktické funkční ukázce, kterou si uživatel může sám pustit a pochopit, jaký význam v zobrazování má dynamický rozsah. První kapitola 2 tedy slouží jako teoretický úvod pro pochopení základních principů. Další kapitola 3 představuje návrh výstupní aplikace, její hlavní komponenty a funkci. Na tento návrh navazuje další kapitola 4 o způsobu implementace této aplikace. Na závěr je uvedena pasáž o výsledcích a testování 5.

Práce je poskytnuta pod licencí svobodného softwaru jako zdroj pro studium vykreslování obsahu ve vysokém dynamickém rozsahu v API Vulkan a může být základem pro další vědeckou práci a vývoj. Zveřejnění proběhlo na internetové stránce *Github* a její snímek obrazovky je k dispozici na konci tohoto dokumentu v příloze B.

Kapitola 2

Problematika zobrazování 3D scén ve vysokém dynamickém rozsahu

Na rozdíl od zobrazování statických obrazů v oboru fotografie je problém vysokého dynamického rozsahu (označován napříč jazyky zkratkou HDR[1]) ve virtuálním trojrozměrném prostoru s výpočtem v reálném čase mnohem komplexnější a náročnější. Zahrnuje kromě mapování tónů a efektů následného zpracování také otázku způsobu ukládání textur v interní struktuře aplikace, návrh percepčních algoritmů simulujících dojem reálného chování lidského zrakového systému a velmi důležitý výkon všech použitých technik s dopadem na celou aplikaci. Podstatné je také zmínit způsob zobrazování HDR scén, a to s ohledem na současné obrazovky a jejich budoucnost.

V této kapitole jsou vysvětleny a popsány všechny výše zmíněné důležité části, ze kterých se renderování[2] 3D scény v HDR skládá a jejich souvislosti. Jako první následuje uvedení do historického kontextu.

2.1 Vysoký dynamický rozsah

Technika HDR má kořeny v oboru fotografie. Obecný základní koncept HDR obrazů spočívá v tom, že jeden snímek obsahuje současně detaily velmi tmavých i velmi světlých míst – typický příklad je fotografie s detaily mraků na obloze a zároveň s detaily krajiny pod oblohou. V tomto pojetí je důležitá tzv. expozice. Zjednodušeně se jedná o dobu, po kterou se při stisku spouště fotoaparátu nechá dopadat světlo ze scény na snímek fotografického filmu (nebo na obrazový snímač v případě digitálního přístroje). S různou dobou expozice se tak dají pořizovat snímky různé podoby stejné scény. To umožňuje mimo jiné práci s pohybem, ale také variabilní kontrast scény. Při krátké době expozice jsou zachyceny detaily míst s vysokou intenzitou světla a při dlouhé expozici jsou naopak zviditelněna místa tmavá. Následným skládáním různě exponovaných snímků do jednoho pak lze dosáhnout obrazu s vysokým dynamickým rozsahem.

Tuto skutečnost pochopil jistý Gustav Le Gray již v 50. letech 19. století a jeho snímky jsou tak považovány za vůbec první HDR fotografie na světě. Snažil se vyřešit problém extrémně kontrastních venkovních scén zachycujících současně oblohu i mořskou krajinu. Zjistil, že pokud pořídí dva negativy, jeden s expozicí oblohy a druhý s expozicí krajiny a oba negativy zkombinuje, získá výsledek s rozšířeným dynamickým rozsahem [21]. Jedna

[1] High dynamic range (HDR) – anglická zkratka pro vysoký dynamický rozsah.

[2] Proces, při němž ze zadaných dat vzniká cílový obraz počítačové grafiky.

z těchto fotografií krajiny je vidět na obrázku 2.1. Tato technika se později začala označovat jako tzv. vícenásobná expozice nebo také skládání expozice [3].

Obrázek 2.1: „Brig upon the Water“ – Gustave Le Gray, kolem roku 1856.[3]

Později se vyvinuly další metody, které napomohly k rozvoji HDR obrazů. První a nejvýznamnější byla tzv. *Dodging and Burning*[4]. Jedná se o techniky z temné komory umožňující buďto zesvětlit nebo ztmavit libovolnou část snímku jednoduše tím, že se světlo ze zvětšovacího přístroje nechá působit jen tam, kde má být snímek zesvětlen a zbytek snímku je nějakým způsobem zakryt [6]. Tyto techniky rovněž přežily dodnes, a to i v podobě softwarových nástrojů dostupných v různých grafických programech.

Dalšímu vývoji napomohl výrazně také filmový průmysl nebo armáda. Po druhé světové válce začal americký fotochemik Charles Wyckhoff spolupracovat s oddělením amerických vzdušných sil na vývoji techniky pro fotografování atomových explozí. Takové exploze produkují enormní množství světelných paprsků s velkým rozdílem v intenzitách. Wyckhoff tento problém vyřešil speciálním filmem, jehož poměr nejmenšího a nejvyššího jasu dosahoval $1 : 10^8$. Koncem 50. let 20. století pak byl publikován první detailní snímek výbuchu termonukleární bomby[5].

Pro grafické 3D aplikace byl nejdůležitější až konec 20. století a přelom tisíciletí. Prvním impulzem bylo vyvinutí Radiance RGBE (formát ukládání obrazu se sdíleným exponentem)[6] archivačního formátu HDR snímků v roce 1985 Gregem Wardem a pokračovalo mnoha dalšími pracemi od výzkumníků jako Paul Debevec (metoda složení různě exponovaných snímků do jednoho HDR obrazu [9], 1997), Erik Reinhard (velmi jednoduchý ale výkonný operátor mapování tónů [25], 2002), M.D. Fairchild a G.M. Johnson (model vzhledu barev iCAM [12], 2002), F. Drago a K. Myszkowski (adaptivní logaritmické mapo-

[3]Převzato z wikipedia.org – článek o Gustavu Le Grayovi. Public Domain.

[4]`https://www.alexbond.com.au/burning-dodging-darkroom-tools/`

[5]Fotka k nahlédnutí např. zde: `https://commons.wikimedia.org/wiki/File:Ivy_Mike_-_fireball.jpg`

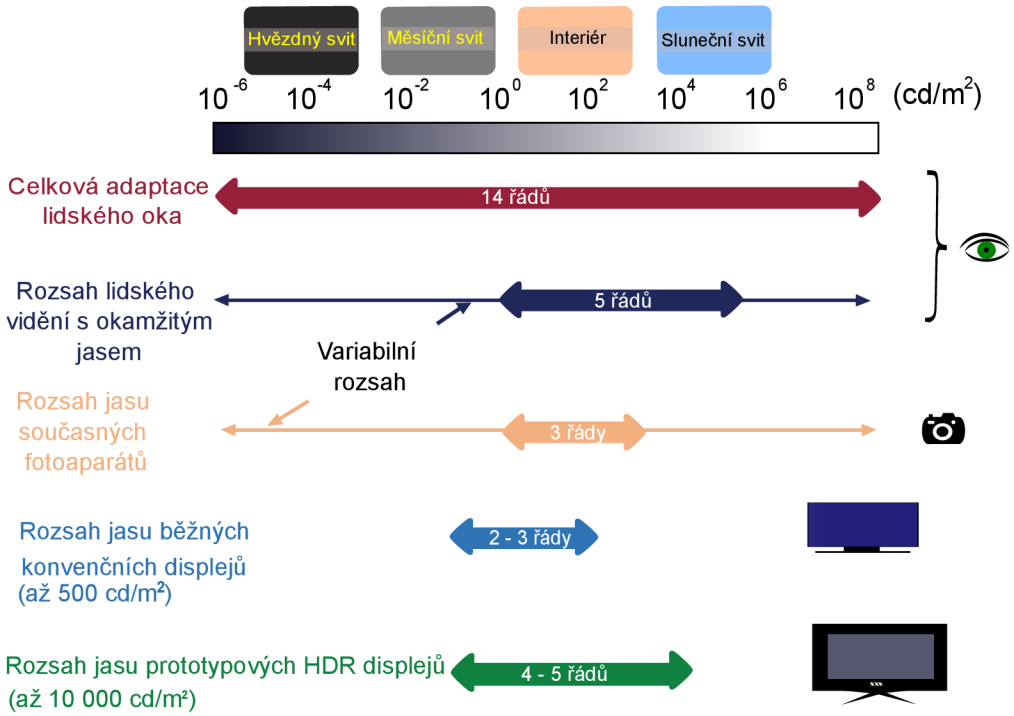
[6]Red Green Blue Exponent (RGBE): `https://www.graphics.cornell.edu/~bjw/rgbe.html`

vání [10], 2003) a dalších. Některé z prací těchto autorů budou více přiblíženy v kapitole o mapování tónů 2.5.

2.1.1 Dynamický rozsah

Rozeznávání detailů v různě světlých/tmavých částech snímku je problém rozsahu jasů, který se odborně popisuje termínem „dynamický rozsah“. Ten je možné z obecného hlediska definovat jako rozdíl největšího a nejmenšího bodu oboru hodnot určité veličiny. V kontextu obrazových dat, jejich zpracování a reprodukci jde o rozdíl mezi největší a nejmenší hodnotou jasu [5]. Jas se v tomto případě označuje jako svítivost a je udáván v jednotkách SI jako kandela na metr čtvereční – cd/m^2.

Dynamický rozsah reálného prostředí je vidět na obrázku 2.2. Lidské oko není schopné vidět v celém rozsahu najednou, ale dokáže se v tomto rozsahu pohybovat a adaptovat na konkrétní úseky, jak znázorňuje užší modrá šipka. Lze snadno vidět, že současná snímací ani zobrazovací technika zatím nepokrývá ani polovinu dynamického rozsahu reality. Označení, že obraz je ve vysokém dynamickém rozsahu proto nutně neznamená, že pokrývá celý rozsah reálného světa. Tento problém více vysvětluje následující podkapitola 2.1.2.



Obrázek 2.2: Srovnání dynamického rozsahu reálného světa s rozsahy lidského zrakového systému, běžných snímacích zařízení a zobrazovacích zařízení. Převzato a upraveno [5].

2.1.2 Vysvětlení HDR v souvislosti se zobrazováním

Scénu nebo snímek lze označit zkratkou HDR tehdy, pokud je jejich dynamický rozsah jasů podstatně vyšší než dynamický rozsah zobrazovacího nebo snímacího zařízení. Označení HDR je tedy relativní a není vyjádřené žádným jedním konkrétním číslem nebo exaktní

definicí. Z toho důvodu ani nelze říci, že je nějaká scéna/snímek více či méně HDR. Nelze ani tvrdit, že se nějaká scéna/snímek více či méně blíží skutečnému HDR. Standardy sice dnes již existují, ale jsou specifikovány výrobci zobrazovacích zařízení podle vlastních kritérií a každá certifikace se v jistých parametrech liší. Více o tom podkapitola 2.2.

Opakem je nízký dynamický rozsah (dále LDR[7]), který je výsledkem ztrátového procesu od snímání po zobrazování. K první ztrátě obrazových a jasových dat dochází už při snímání, kdy zařízení snímá v menším rozsahu, než jaký má daná scéna. I když je už v současnosti technologie snímání na velmi vysoké úrovni a ztráta obrazových dat téměř minimální, degraduje reprodukci obrazu dále také následné zpracování, ale hlavně poslední fáze na konci procesu, tedy samotná vizualizace prostřednictvím zobrazovacích zařízení.

Jak ukazuje obrázek 2.3, LDR zobrazení v určitých typech scén silně zkresluje původní obraz. Je to dáno tím, že SDR[8] obrazovka je schopna interně pracovat pouze s omezeným rozsahem hodnot barev. Původní hodnoty z reálné scény v některých částech několikanásobně překračují tento nativní rozsah obrazovky, která všechny hodnoty mimo rozsah mapuje na stejnou maximální (nebo minimální) hodnotu. Výsledkem jsou přesycené plochy, kde každý pixel vyzařuje stejnou hodnotu (v případě obrázku 2.3 maximální možný odstín bílé) a všechny detaily v této oblasti jsou ztraceny. Techniky HDR zobrazování poskytují řešení a jejich úkolem je upravit intenzity barev v celém obrazu tak, aby zachovaly zároveň v jedné scéně detaily velmi tmavých i silně přesvícených oblastí.

Pro plnohodnotné zobrazení s vysokým dynamickým rozsahem jsou dnes nezbytné tři věci – hardware (dedikovaná nebo integrovaná grafická karta) s ovladači podporující HDR vykreslování, obsah vykreslený v HDR a zobrazovací zařízení podporující zobrazení HDR obsahu. Následující podkapitola se věnuje právě posledním zmiňovaným, tedy obrazovkám.

2.2 Zobrazovací zařízení

Dnes již obrazovky s funkcí HDR sice nejsou tak vzácné a nedosažitelné, ale stále nejsou rozšířeným spotřebitelským standardem napříč všemi uživateli a domácnostmi[9]. Některé z mnoha důvodů jsou technologická náročnost výroby nebo vysoká spotřeba elektrické energie. Hlavním problémem z pohledu 3D grafiky je však náročnost vykreslování HDR obsahu. Ne každá grafická karta si v HDR poradí se všemi druhy 3D aplikací a i proto se stále ještě po několika dekádách nepodařilo implementovat vysoký dynamický rozsah do všech zobrazovacích zařízení a v nejbližší době nejspíš ani nepodaří. Z toho důvodu je velmi důležitým cílem při renderování navrhovat algoritmy s co možná nejnižší zátěží na paměť a výkon grafických jednotek. Více o tom ale pojednává podkapitola o vykreslování v HDR 2.3. Následující text se věnuje důležitým parametrům a vlastnostem zobrazovacích zařízení, které významně ovlivňují výsledný obraz HDR.

2.2.1 Kontrastní poměr

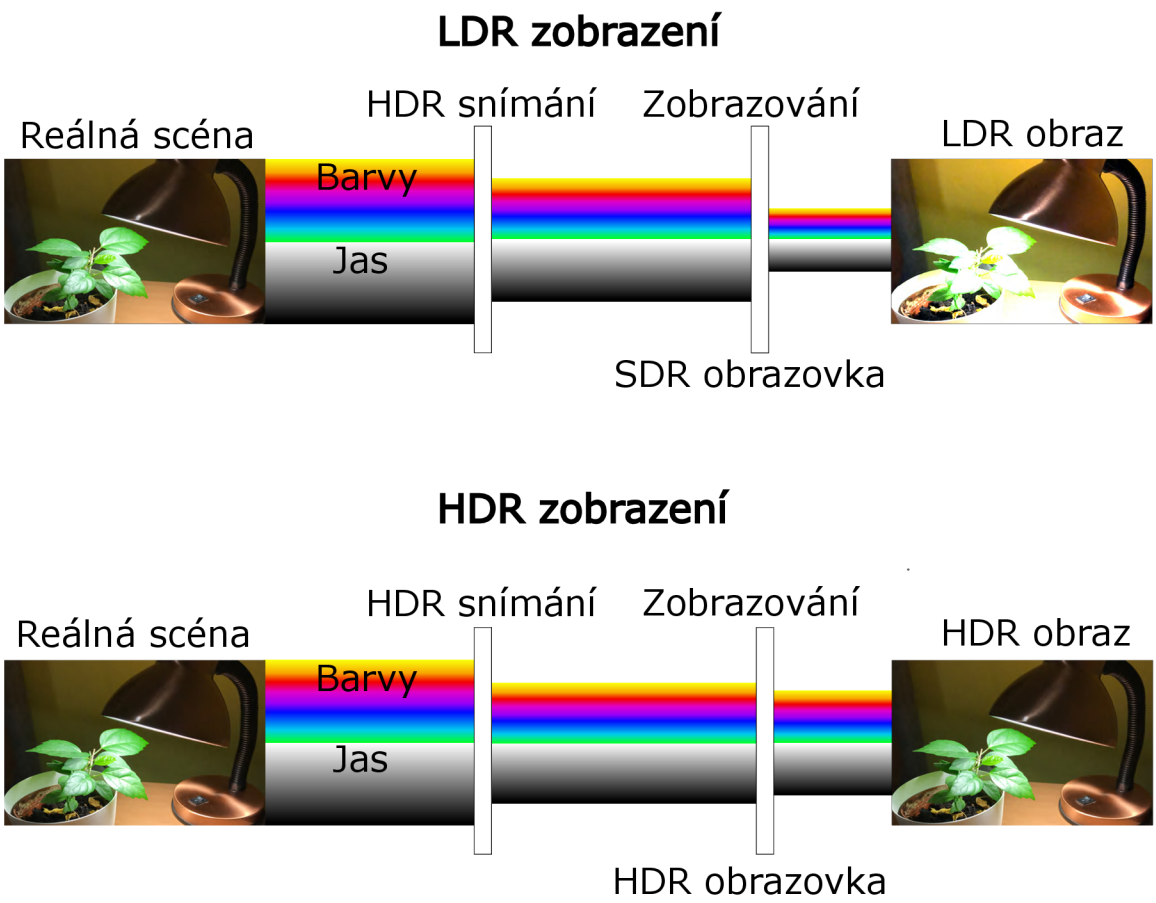
Nejen v komerční oblasti se pro vyjádření dynamického rozsahu často používá pojem kontrastní poměr[10], který udává poměr jasu nejtmavší a nejjasnější barvy, kterou je zařízení schopno zobrazit [5]. U dnešních běžných konvenčních LDR monitorů dosahuje hodnoty kolem 300:1 a u obrazovek se schopností HDR až 10 000:1. To už je přibližná hodnota

[7] Low dynamic range (LDR) – nízký dynamický rozsah.

[8] Standard dynamic range (SDR) – standardní dynamický rozsah (jiný výraz stejného významu pro LDR).

[9] Pro představu cenové dostupnosti: `https://www.displayninja.com/hdr-monitor-list/`.

[10] Contrast ratio (CR) – kontrastní poměr.



Obrázek 2.3: Významnost vysokého dynamického rozsahu v porovnání s nízkým rozsahem. Vlevo vždy scéna zachycená kamerou schopna snímat v HDR a vpravo je tento snímek zobrazen na SDR nebo HDR displeji.

rozsahu, který rozeznává lidské oko po adaptaci, jak je znázorněno na obrázku 2.2. Jenže lidský zrakový systém má tento adaptační rozsah variabilní a dokáže se přizpůsobovat jasu v reálné přírodě v rozsahu až čtrnácti řádů. Z toho důvodu jsou sice obrazovky s tímto poměrem už značně výkonné, ale vizuálně mnohem kvalitnější HDR výsledek poskytují přístroje s poměrem až 1 000 000:1[11].

Kontrastní poměr je pro dosažení HDR velmi důležitý, některé zdroje uvádí doporučené minimální hodnoty, které by měly zaručovat kvalitní zobrazovaný výsledek. Pro maximální tmavý (černý) stav $0,01nit$[12] (a méně) a pro maximální jasový (bílý) stav alespoň $1000nit$ (a více), což určuje efektivní kontrastní poměr nejméně 100 000:1 [7]. Dosažení hodnoty menší než $0,01nit$ je pro LCD displeje velmi náročné. Pro zlepšení tohoto nedostatku byla vyvinuta technika lokálního stmívání.

2.2.2 Lokální stmívání

Pro renderování v HDR je lokální stmívání velmi důležitý pojem, který má velký potenciál, ale je velmi náročný na technologickou realizaci. Tzv. *local dimming* je jeden ze způsobů,

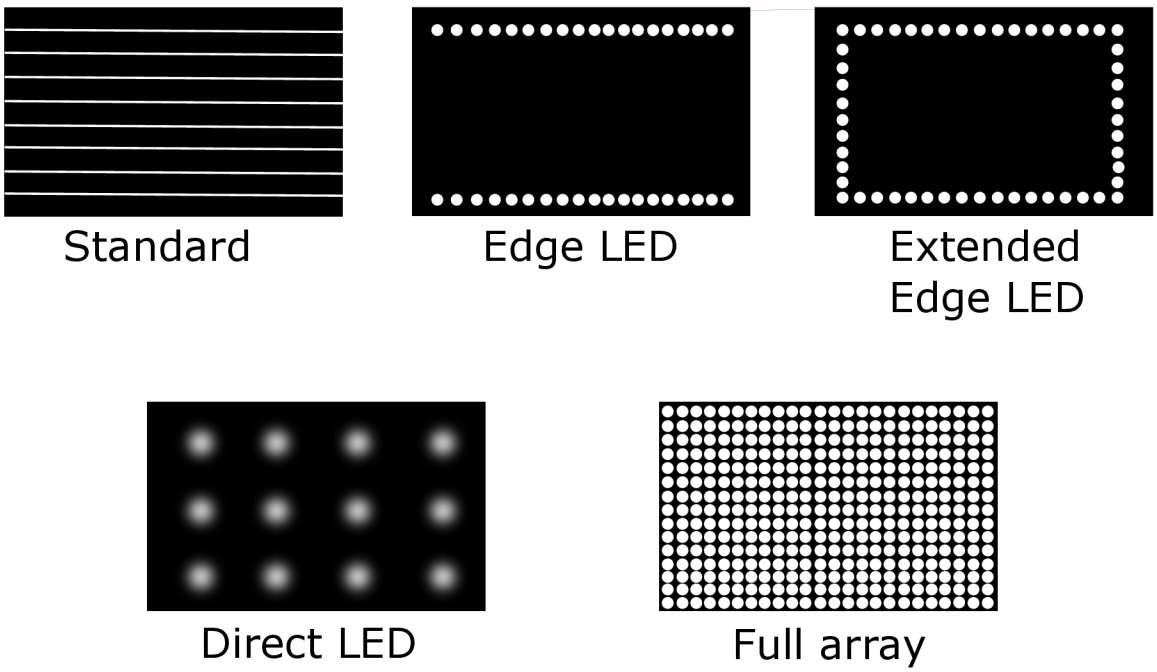
[11] `https://www.digitaltrends.com/computing/pc-gaming-hdr-problem-a-way-out/`
[12] Nit - jiný termín pro cd/m^2 vyjadřující stejnou jednotku.

kterým se dá u LED LCD obrazovek dosáhnout obrazu s vysokým dynamickým rozsahem, zkvalitnit výsledný vyrenderovaný HDR obsah a poskytnout lepší subjektivní vizuální dojem pro pozorovatele.

Jedná se o techniku rozdělující LED podsvícení LCD monitorů na malé segmenty. V závislosti na obsahu, který má být zobrazen, se dané segmenty (zóny) buď ztmaví nebo zesvětlí a tím zdánlivě zvýší celkový kontrastní poměr. Například zóny podsvicující ty části obrazu, na kterých je temný objekt ve stínu, se ztmaví, a naopak zóny v oblastech, kde zrovna ve scéně jasně svítí nějaký zdroj světla, zvýší svou intenzitu podsvícení. Tady ovšem naráží celý princip na technologické limity, jak již bylo zmíněno v úvodu této podkapitoly. Počet segmentů je zcela rozhodujícím faktorem pro kvalitní výsledek. V současné době se počet zón u konvenčních obrazovek pohybuje v desítkách až stovkách jednotek, u dražších panelů to jsou až dvě tisícovky[13]. To sice není malý počet, ale lokálnímu stmívání na úrovni pixelů, které zvládají OLED displeje, se bude zónové podsvícení LCD panelů vyrovnávat velmi těžko. Počet segmentů by musel vzrůst na několik desítek až stovek tisíc, což současné technologie zatím nedovolují. Systém rozložení LED zón ilustruje obrázek 2.4.

Problémy zónového podsvícení

Počet a rozložení zón je velmi podstatné pro eliminaci obrazových artefaktů, které lokální stmívání může vykazovat. Například tzv. „*blooming*“ nebo též „*halo*“ efekt – pokud je daná zóna osvětlena, může sousední neosvětlenou zónu ovlivnit a vznikne mezi nimi nežádoucí jemný přechod světla a objekt ve scéně se může jevit kolem okrajů rozmazaný. Další vedlejší efekt může nastat v případě, kdy je objekt ve scéně (například hvězda na noční obloze) menší než daná zóna. V takové situaci se zóna nemusí spustit a objekt se ve výsledku jeví jako tlumený a nevýrazný – což je přesný opak smyslu HDR.



Obrázek 2.4: Způsoby rozvržení zónového podsvícení.

[13]Neúplný výčet zařízení s uvedením počtu zón: `https://www.displayninja.com/hdr-monitor-list/`

Nejlepších výsledků dosahují obrazovky s tzv. *full-array* lokálním stmíváním pro jejich hustotu zón a nejméně vykazovaných obrazových artefaktů. Vysoký počet segmentů však zvyšuje náklady na výrobu a tím i pořizovací cenu.

Lokální stmívání je tak obecně dobrý krok k dosažení velmi vysokého dynamického rozsahu LCD displejů a v ideálních podmínkách bez artefaktů výsledek HDR renderingu věrně reprodukuje. Bohužel provedení této techniky je zatím značně limitováno dostupnými technologiemi a vykazování artefaktů je tak poměrně časté.

Lokální stmívání na úrovní pixelů

Protože neustálé zvyšování počtu zón má své limity, začali v posledních letech výrobci televizorů vyvíjet novou technologii s názvem *Dual-cell LCD* [7]. Princip spočívá v použití dvou LCD matric řazených podle pixelů 1:1 precizně za sebou. Primární vrchní matrice ovládá barevné odstíny pixelů (tedy stejným principem jako pracuje standardní LCD matrice) a sekundární má pouze černobílé pixely, pracuje tedy pouze s jasem v odstínech šedi. Tlumí pixely při zobrazování tmavých odstínů, tzn. že při zobrazení černého bodu se zablokuje nejen barevný primární pixel, ale i sekundární černobílý a lépe tak tlumí zdroj světla z podsvícení.

Důležitou technologií představující budoucnost dual-cell LCD je tzv. OLCD[14], která funguje na tenké fólii, což v porovnání se sklem používaným u LCD displejů dovoluje vrstvit matrice mnohem těsněji na sebe.

2.2.3 Množiny barev a jejich reprezentace

Dalším parametrem, který je důležitou součástí HDR standardů a výrazně ovlivňuje vizuální kvalitu HDR obsahu, je barevný prostor a gamut. Čím širší jsou tyto množiny hodnot, tím kvalitnější je výsledný obraz. Základem těchto prostorů je barevný model.

Barevný model

Barevný model je matematický model popisující viditelné barevné spektrum reálného světa a zobrazuje barvy v něm obsažené jako vícerozměrný model. Díky číselné reprezentaci je barvy možno počítačově zpracovávat a zobrazovat. Většina barevných modelů má tři složky (tři barvy), jsou tedy třírozměrné a lze je zobrazit jako 3D obrazce[15].

Pro renderování v HDR je stěžejní model RGB (červená, zelená, modrá) široce využívaný v počítačové grafice a programování softwaru, kvůli snadné reprezentaci pomocí bitů a jednoduchosti míchání barev. Existují ale i modely jako CMYK (azurová, purpurová, žlutá a černá) sestávající ze čtyř složek, využívaný zejména pro tisk.

Velmi názorná interaktivní grafická ukázka, jak pracují různé barevné modely v prostoru sRGB, je dostupná k vyzkoušení na výukových stránkách o grafice autora Rune Madsena[16].

Barevné modely definují množiny barev, které se nazývají barevné prostory.

Barevný prostor a gamut

Barevný prostor je množina všech barev, spadající do množiny daného barevného modelu, které je možné určitým způsobem snímat, zpracovávat a zobrazovat. Jinými slovy tento pro-

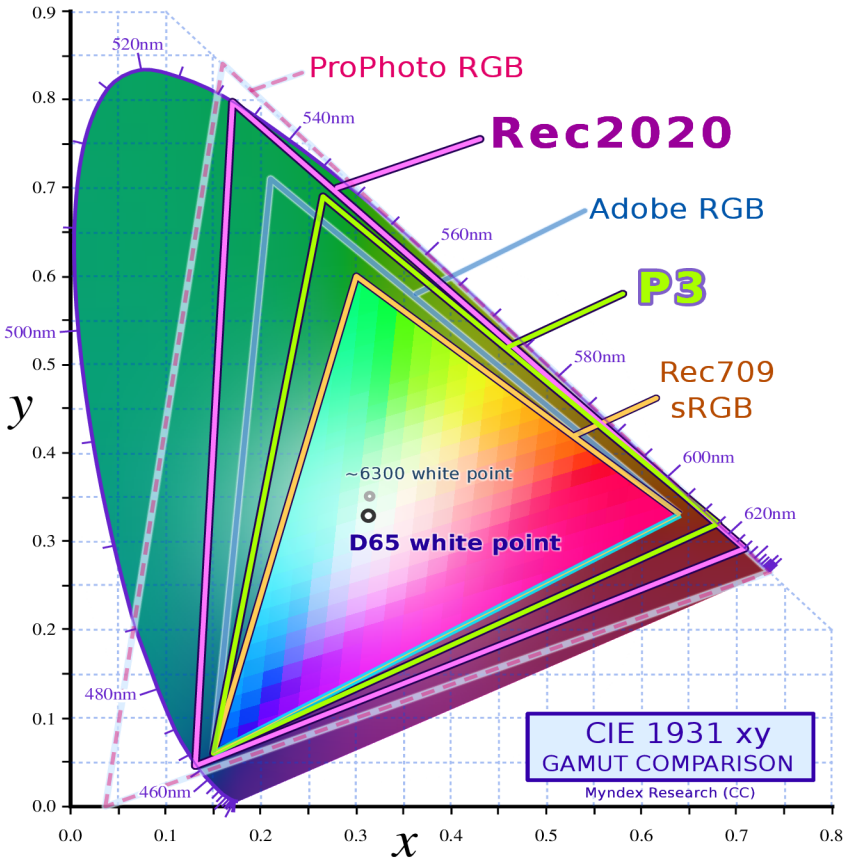
[14] Organic LCD (OLCD).

[15] `https://www.sciencedirect.com/science/article/pii/S1049965283710199`

[16] `https://programmingdesignsystems.com/color/color-models-and-color-spaces/index.html`

stor definuje určitou část barev z viditelného spektra a těmto jednotlivým barvám přiřazuje číselné hodnoty způsobem, který definuje barevný model, ze kterého prostor vychází.

Barevný gamut pak označuje rozsah barev z barevného prostoru, který je zobrazovací zařízení schopno zpracovávat a reprodukovat. Často se tyto pojmy nesprávně zaměňují, ovšem nejsou si rovny.



Obrázek 2.5: Porovnání standardizovaných prostorů proti diagramu chromatičnosti.[17]

Na obrázku 2.5 je znázorněn diagram chromatičnosti jako zakřivený podkovovitý obrazec. Vytvořený byl Mezinárodní komisí pro osvětlení (CIE)[18] v roce 1931. Diagram s označením „CIE 1931 xy“ vyjadřuje pouze barvu bez jasové složky. Je vyobrazením barevného prostoru pokrývající kompletně všechny barvy viditelné lidským okem. Trojúhelníky uvnitř tohoto prostoru jsou barevné prostory, ve kterých jsou schopné pracovat dnešní digitální snímače, zobrazovače a zařízení tisku.

Tyto prostory lze rozdělovat podle modelů, na kterých jsou založeny (na obrázku figurují pouze prostory barevného modelu RGB).

Prvním prostorem je sRGB[19] (Rec709), který byl vytvořen v roce 1996 a následně standardizován Mezinárodní elektronickou komisí IEC[20]. Ve své době CRT obrazovek zcela

[17] Převzato z wikipedia.org – článek o specifikaci Mezinárodní telekom. unie Rec.2020. CC BY-SA 4.0.
[18] `https://cie.co.at`
[19] Standard RGB (sRGB) – vytvořený ve spolupráci firem HP a Microsoft v 90. letech.
[20] Standard IEC 61966-2-1:1999: `https://webstore.iec.ch/publication/6169`.

vyhovoval, avšak pro dnešní vykreslování v HDR je poměrně nevhodný. Poskytuje sice obrazy vcelku s věrným podáním barev, ale pro dosažení maximálně kvalitních výsledků jsou vhodnější prostory s širším rozsahem, jako je například prostor definovaný sadou doporučení vydanou ITU[21] BT.2020 [28] (nebo rozšířené doporučení zaměřené na produkci HDR obrazů BT.2100 [29]), které nabídnou více barev a lidské oko obraz považuje za mnohem realističtější.

Barevný prostor s označením Rec.2020 (BT.2020), jak je vidět na obrázku 2.5, pokrývá mnohem více barev z reálného prostředí a je tím pádem velmi vhodný pro reprodukci HDR obsahu téměř beze ztráty barevné informace. Má však jeden technický nedostatek – v současné době je zatím velmi obtížným úkolem vyrobit zobrazovací zařízení s gamutem pokrývajícím 100 % tohoto prostoru. Většina dostupných obrazovek dosud zvládá maximálně kolem 90 %.

Při implementaci HDR renderovacích algoritmů je důležitá namísto gamutu především bitová hloubka, tedy počet všech možných barev jednoho pixelu v závislosti na počtu bitů reprezentujících barvu jednoho pixelu. Více tento parametr popisuje kapitola 2.3.

2.2.4 HDR standardy

Označení HDR je u displejů velmi relativní, protože certifikací specifikujících parametry tohoto označení je více. Různí výrobci deklarují různé specifikace, nejrozšířenějšími jsou *DisplayHDR*, *HDR10*, *HDR10+* a *Dolby Vision*. Obecně tyto standardy nevyjadřují jen dynamický rozsah, ale zahrnují také požadavky na rozlišení obrazovky, bitovou hloubku, minimální hodnotu maximálního vyzařovaného jasu, maximální hodnotu minimálního vyzařovaného jasu ve stavu plně černého pixelu a barevný gamut.

2.3 Renderování ve vysokém dynamickém rozsahu

Vykreslování v HDR obecně sestává z několika komponent, jejichž řazení je popsáno na obrázku 2.6. Zobrazovací část je vysvětlena v předcházejících podkapitolách a pracuje nezávisle na vykreslovací části, ačkoli v závislosti na typu zobrazovacího zařízení se renderovací segment musí rozhodnout, jaký přístup použije. Následující text této podkapitoly popisuje hlavní složky, které tvoří proces renderování 3D prostředí ve vysokém dynamickém rozsahu.

2.3.1 Lidský zrakový systém

Při vykreslování virtuálního 3D světa je hlavním cílem dosáhnout co největšího pohlcení tímto virtuálním prostředím a vtáhnutí diváka do děje. Úkolem je tedy co nejvyšší míra realističnosti v každém směru, tzn. precizně modelované objekty, skutečně se jevící celkový pohled na scénu ve smyslu osvětlení včetně nedokonalostí lidského vidění a reálné chování všech objektů ve scéně, a to i včetně kamery jejíž chování by mělo odpovídat lidskému zrakovému systému.

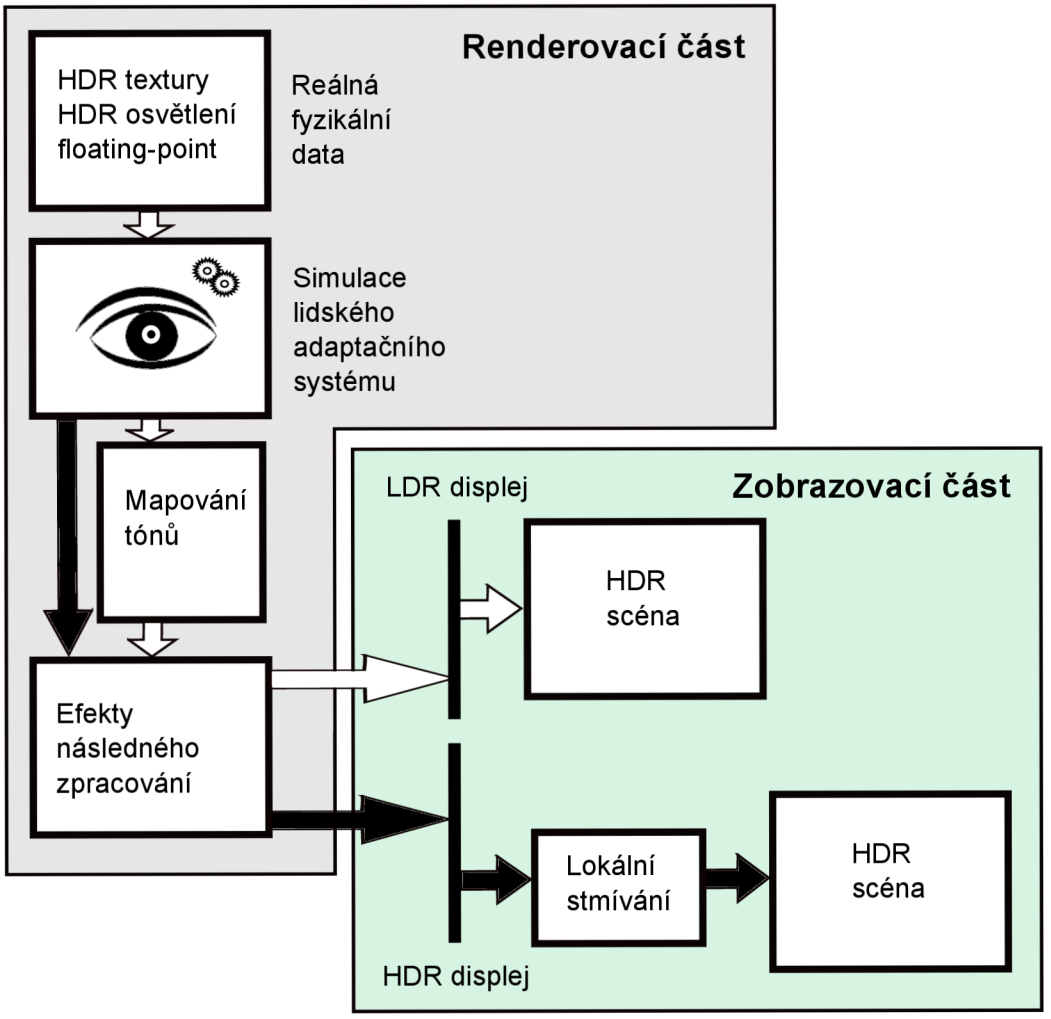
Podle přístupu, který navrhl J. A. Ferwerda [13], je potřeba pro dosažení realistické podoby scény splnit tři hlavní kritéria – splnění fyzikálního realismu, fotorealismu a funkčního realismu.

První jmenované kritérium požaduje, aby obraz poskytoval fyzikálně přesný popis osvětlení scény a s tím související přesný popis tvarů a materiálů objektů pro správný odraz nebo pohlcení světla.

[21] International Telecommunication Union (ITU).

Druhé kritérium v podstatě vyžaduje simulaci adaptačního systému lidského oka, které se tímto zvládne přizpůsobovat celému rozsahu světelných intenzit v přírodě.

A konečně poslední ze seznamu je funkční realismus, který zakládá na myšlence, že objekty a interakce s nimi musí odpovídat reálnému chování, které by se očekávalo ve skutečném světě.



Obrázek 2.6: Úplný proces zobrazování ve vysokém dynamickém rozsahu.

2.4 Techniky a efekty HDR

S ohledem na smysl těchto doporučení je nutné, aby HDR renderer[22] byl sestaven z částí, které jsou vidět na obrázku 2.6 v šedém ohraničení.

Reálná fyzikální podstata objektů ve scéně je zajištěna použitím textur v některém z HDR formátů (například Radiance RGBE nebo OpenEXR), popis osvětlení provést na-

[22]Program nebo algoritmus, který vykresluje objekty poč. grafiky do podoby zobrazitelné na obrazovce.

příklad pomocí tzv. Phongova osvětlovacího modelu nebo využít tzv. *ray tracing*[23] a veškerá data aplikace ukládat v plovoucí řádové čárce alespoň s šestnáctibitovou přesností.

O fotorealismus se stará dvojice segmentů. Simulace lidského adaptačního systému může být realizována jako funkce automatické expozice, která zjistí průměrnou hodnotu jasu v aktuální scéně, přepočítá expozici podle vzorce a nastaví ji na nově vypočtenou hodnotu. Příkladem může být přechod kamery z temné místnosti do venkovního prostředí s přímým slunečním svitem, protože pro vnitřní prostředí je třeba mít vyšší expozici než venku. Mezi další fotorealistické efekty patří například různé účinky oslnění silným zdrojem světla. Typickým příkladem je tzv. *glare* efekt, který dokáže navodit dojem rozmazaného okolí zdroje oslnění nebo paprsků vyčnívajících ze zdroje tohoto silného světla. Podobných efektů simulujících nedokonalosti zrakového systému je celá řada a spadají do tzv. *post-processing* úprav.

Splnění podmínky funkčního realismu závisí na charakteru dané 3D aplikace.

2.5 Mapování tónů

Aby mělo vykreslování v HDR smysl, je třeba výslednou vyrenderovanou scénu adekvátně zobrazit. Takové adekvátní zobrazování je popsáno v předcházejících kapitolách o obrazovkách a společně s renderovací částí tvoří úplný průběh vykreslování ve vysokém dynamickém rozsahu. V současnosti jsou však výrazně rozšířenější stále ještě LDR displeje a jak je vidět na obrázku 2.6, je pro ně nutné zvolit odlišný přístup. Tento přístup spočívá v aplikování tzv. „mapování tónů“, které má za úkol určitou funkcí redukovat dynamický rozsah HDR snímku na nižší dynamický rozsah LDR snímku zobrazitelný na LDR obrazovce. Podrobně se tomuto mapování věnuje tato podkapitola.

I když mají LDR obrazovky drtivou převahu v rozšířenosti, je mnohem vhodnější snímat a renderovat scény v HDR a následně je tónově mapovat pro zobrazení na LDR displejích, než snímat a vykreslovat scény přímo v LDR a zobrazovat bez mapování. Scény vzniklé s vysokým dynamickým rozsahem poskytují na LDR displejích po mapování mnohem kvalitnější výsledné obrazy [19].

Operátory mapování tónů

Operátory mapování tónů označované jako tzv. TMO[24] nebo TMOs[25] pro vyjádření množného čísla, jsou algoritmy pro mapování tónů, které mohou mít v závislosti na konkrétní aplikaci různé cíle. Některé TMOs například fungují pouze jako estetické filtry, jiné se snaží o reprodukci maximálního počtu velmi jemných detailů v obrazu, další se zase snaží zacílit na maximalizaci kontrastu. Jiným typem mohou být operátory, které jsou navrženy tak, aby generovaly LDR obrazy percepčně shodné s původní scénou s vysokým dynamickým rozsahem (obecně zaměřeny na aplikace realistického vykreslování).

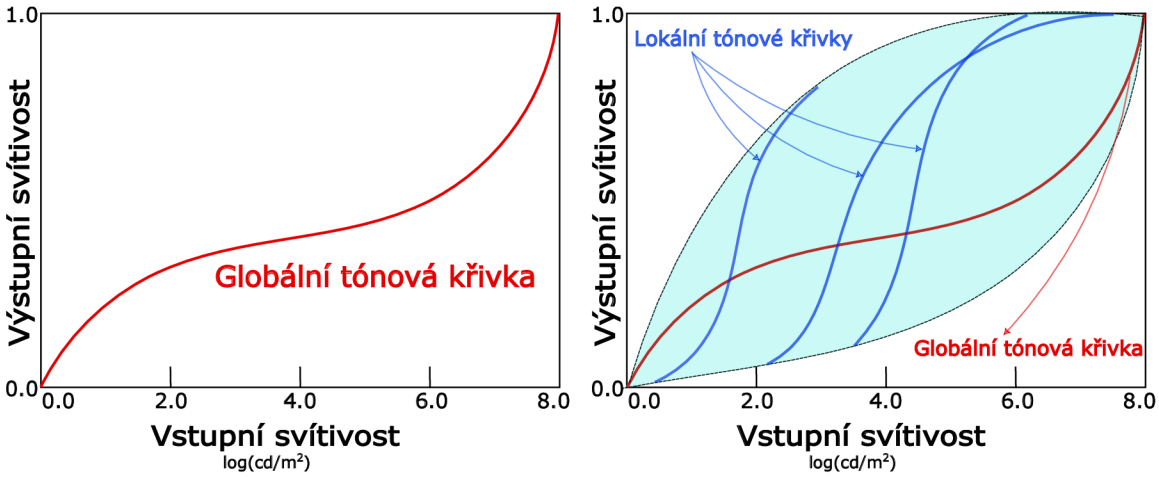
Koncept mapování tónů poprvé představili Tumblin a Rushmeier v roce 1993 [27] a od tohoto data vzniklo velké množství dalších TMOs. Pro orientační představu, k dnešnímu dni vyhledávač Google Scholar nalezl pro vstupní řetězec *tone mapping operator* 147 000 výsledků, které se vztahují k vědeckým nebo technickým publikacím zabývající se nějakým způsobem operátory mapování tonů. Je patrné, že zájem o vývoj a evaluaci těchto operátorů je velký.

[23] Metoda glob. osvětlení, která počítá paprsky osvětelní pomocí drah vedených z kamery pozorovatele.
[24] Tone mapping operator (TMO) – operátor mapování tónů.
[25] Tone mapping operators (TMOs) – operátory mapování tónů.

Z důvodu jejich velkého počtu je vhodné je pro větší přehlednost členit do kategorií podle společných vlastností. Nejzákladnější dělení je možné na globální a lokální TMOs:

- **Globální TMOs**
  Často nazývané také jako prostorově invariantní aplikují na všechny pixely obrazu stejnou funkci (jednu mapovací křivku) nezávisle na vlastnostech okolních pixelů. Jedna vstupní hodnota vede k jedné a pouze jedné výstupní hodnotě. Globální operátory jsou ze své podstaty obecně jednodušší na implementaci a současně méně náročné na výpočetní čas, ale mohou snižovat lokální kontrast. Pro jejich vysoký výkon jsou nadmíru vhodné pro použití v real-time aplikacích.

- **Lokální TMOs**
  Jiným názvem prostorově variantní[26], tzn. že parametry funkce se v každém pixelu mění na základě vlastností okolního prostoru. Tyto algoritmy jsou podstatně složitější a vlivem toho i výrazně náročnější na výkon. Mohou vykazovat typické obrazové artefakty a výsledný obraz se může jevit až nerealisticky, ovšem s velmi výraznými detaily.



Obrázek 2.7: Křivky globálních a lokálních TMO.

Obrázek 2.7 ukazuje rozdíl mezi přístupem globálních a lokálních operátorů. Vlevo je křivka globálního mapování tónů, která je prostorově neměnná a pro každý bod v obraze stejná. Naproti tomu lokální přístup vpravo využívá různých průběhů mapování v závislosti na poloze bodu v obraze.

Další selekce mohou být různorodé. Například podle cílů, kterých má daný TMO dosáhnout, tedy percepčně přesná reprodukce, věrná reprodukce barev a reprodukce podle subjektivních záměrů [56]. Dále lze dělit podle toho, zda se TMO zaměřuje pouze na mapování jasu nebo pokrývá i přesnou reprodukci barev [18]. V takovém případě je operátor založený na tzv. *Image color appearance model*[27] (TMO označovaný jako iCAM [12]).

Významnou skupinu operátorů tvoří ty, které principiálně vychází z vizuálního modelu, tedy z chování a vlastností lidského zrakového systému (anglická zkratka HVS[28]), který

[26]Prostorová variabilita – hodnoty veličiny měřené v různých místech určitého prostoru se v jednotlivých místech liší.
[27]Model barevného vzhledu obrazu.
[28]Human visual system (HVS) – volně přeloženo jako lidský zrakový systém.

uvádějí některé významné publikace [24] jako stěžejní aspekt pro realistické mapování tónů. Jelikož HVS dokáže zpracovat tak obrovský rozsah jasů, jaký se v přírodě vyskytuje, je nanejvýš vhodné inspirovat se tím, jak HVS funguje a aplikovat jeho vybrané vlastnosti v algoritmech mapování tónů. Následuje jeden z nejznámějších TMO.

2.5.1 Operátor fotografické reprodukce tónů

Původním názvem *Photographic tone reproduction* je operátor mapování tónů, který byl uvedený E. Reinhardem v roce 2002 [25]. Výkonově nenáročný, hojně využívaný v aplikacích různých typů a velmi vhodný především v real-time vykreslování. Může být proveden jako globální nebo lokální.

Tato tónová reprodukce čerpá z osvědčených technik oboru fotografie a snaží se o realistické zobrazení snímků. Neusiluje o dokonalé napodobení skutečného fotografického procesu, ale pouze zakládá na tzv. zónovém systému publikovaném již v 80. letech Anselem Adamsem [2]. Tento systém rozděluje scénu podle rozsahů jasu na několik odstupňovaných částí, přičemž jedna zóna je definována jako římská číslice, která spojuje přibližnou odrazivost tisku a přibližný rozsah jasu scény. Pomocí tohoto systému lze dosáhnout řízení volby při reprodukci tónů.

Nejprve se provede obdoba nastavení expozice jako ve fotoaparátu, tedy aplikace globálního škálování obrazu. K tomu je potřeba nastavení „klíče scény“, který je aproximován logaritmickým průměrem jasu $\bar{L}_w$:

$$\bar{L}_w = \frac{1}{N} \exp\left(\sum_{x,y} \log(\delta + L_w(x,y))\right) \tag{2.1}$$

kde $L_w(x,y)$ je jas scény pro pixel (x,y), N je celkový počet pixelů v obraze a δ je malá hodnota, která zabraňuje singularitě při logaritmování v případě, kdy se v obraze vyskytují černé pixely (tedy zabraňuje $\log(0)$). Vypočtená hodnota je následně mapována na hodnotu a definovanou uživatelem podle subjektivních požadavků na jas scény takto:

$$L(x,y) = \frac{a}{\bar{L}_w} L_w(x,y) \tag{2.2}$$

kde a je „klíčová hodnota“ udávající celkový subjektivní jas scény. Světlý obrázek je dán klíčem *high-key*, vyvážený obrázek klíčem *normal-key* a tmavý pomocí *low-key*. Ověřené hodnoty poskytující dobré výsledky jsou $a = 0,18$ pro vyvážené, $a = 0,09$ nebo $a = 0,045$ pro tmavší scény a $a = 0,36$ nebo $a = 0,72$ pro scény se světlým prostředím. Dále se použije operátor globálního mapování tónů, čímž se získají zobrazitelné jasové hodnoty $L_d(x,y)$:

$$L_d(x,y) = \frac{L(x,y)}{1 + L(x,y)} \tag{2.3}$$

kde $Ld(x,y)$ reprezentuje jas displeje. Tato rovnice zaručuje, že hodnoty všech pixelů budou v zobrazitelném rozsahu, což ale může být ještě uživatelsky kontrolováno pomocí parametru, který může záměrně „vypalovat[29]“ nejjasnější oblasti:

$$L_d(x,y) = \frac{L(x,y)\left(1 + \frac{L(x,y)}{L^2_{white}}\right)}{1 + L(x,y)} \tag{2.4}$$

[29]Původ ve fotografii – vypálení daného místa obrazu znamená, že je dané místo absolutně bílé.

kde L_{white} je uživatelem definovatelný parametr. Všechny hodnoty jasu větší, než L_{white} budou namapovány na 1. Pokud je hodnota L_{white} nastavena na maximální (nebo vyšší) jas ve scéně nedojde k žádnému vypalování. Pokud je nastavena na nekonečno, rovnice se zredukuje na rovnici (2.3).

Rovnice (2.4) je jednoduchým globálním operátorem, který dostatečně zachovává detaily oblastí s nízkým kontrastem. Dále zaručuje, že hodnoty jasu všech pixelů budou v zobrazitelném rozsahu 0 až 1. Problém ovšem tento operátor vykazuje v obrazech s velmi vysokým dynamickým rozsahem, kde většinou nedokáže správně reprodukovat jemné detaily, zvláště ve velmi světlých oblastech [25]. Reinhard však navrhl řešení i pro tento případ, a to pomocí algoritmu lokálního zvýšení kontrastu, který je obdobou fotografické techniky *Dodging and burning* (představena již v úvodní podkapitole 2.1).

Pro měření lokálního kontrastu se použije výpočet středového okolí. Nejprve se vypočítá Gaussův vážený průměr pixelu, který reprezentuje střed a poté Gaussův vážený průměr pro stejný pixel ale s větší oblastí, což určuje okolí. Následně se provede rozdíl těchto dvou Gaussových průměrů a pokud je blízký nule, znamená to, že v okolí nejsou žádné výrazné kontrasty. Pokud se však oba průměry výrazně liší, indikuje to kontrastní kraj zasahující do okolí. Gaussův obraz v daném měřítku s je počítán takto [24]:

$$L_s^{blur}(x,y) = L(x,y) \otimes R_s(x,y) \tag{2.5}$$

a mechanismus středu a okolí v daném měřítku je dán funkcí:

$$V_s(x,y) = \frac{L_s^{blur} - L_{s+1}^{blur}}{2^{\phi}a/s^2 + L_s^{blur}} \tag{2.6}$$

kde L_s^{blur} a L_{s+1}^{blur} představují odezvy středu a okolí získány z rovnice (2.5). Výraz $2^{\phi}a/s^2 + L_s^{blur}$ funguje jako normalizace, ve které člen $2^{\phi}a/s^2$ zabraňuje jejímu porušení v případě malých hodnot, kterých by mohl L_s^{blur} nabývat. Parametr a je klíčová hodnota a exponent ϕ je uživatelský parametr, který řídí míru zostření. V případě jeho nastavení na malé hodnoty je účinek takřka neznatelný, naopak při nastavení na příliš velká čísla vykazuje obrazové artefakty *halo*.

Klíčovým využitím rovnice (2.6) je nalezení měřítka s_{max} vhodné velikosti, tj. nalezení největší oblasti kolem daného pixelu, ve které nedochází k velkým změnám kontrastu. Takové měřítko může být jiné pro každý pixel, tzn. že hlavní myšlenkou tohoto Reinhardova vlastního pojetí techniky *Dodging and burning* je způsob výběru správného měřítka s_{max}. Pro výběr největšího okolí se zvolí prahová hodnota V, podle které bude vybráno odpovídající měřítko s_{max} následující podmínkou:

$$|V_{s_{max}}(x,y)| < \epsilon \tag{2.7}$$

kde ϵ je stanovený práh. Výpočet mechanismu středu a okolí se počítá pro různé měřítka s od nejmenšího po s_{max} dokud platí podmínka (2.7). Při správně zvoleném měřítku funguje $L_{s_{max}}^{blur}$ jako lokální plošný jas pro daný pixel a lze jím nahradit L(x,y) ve jmenovateli rovnice (2.3). Z globálního operátoru lze tedy snadno sestavit operátor lokální:

$$L_d(x,y) = \frac{L(x,y)}{1 + L_{s_{max}}^{blur}(x,y)} \tag{2.8}$$

který využívá principu fotografické techniky *Dodging and burning*. Pro jas světlého pixelu v tmavé oblasti bude platit $L > L_{s_{max}}^{blur}$, což způsobí menší kompresi výsledného jasu a pixel bude tím pádem vypálen podobně jako u techniky *burning*. Naopak pro tmavý pixel

ve světlém prostoru bude platit $L < L_{s_{max}}^{blur}$, výsledný jas pixelu L_d bude více snížen a napodobí tím techniku *dodging*. Tento *dodging* a *burning* zapříčiňuje zvýšení kontrastu pixelu vzhledem k okolí, což ve výsledky znamená vyšší detaily v celém obrazu (za předpokladu vhodně zvoleného měřítka). Hodnoty ϕ a ϵ z rovnic (2.6) a (2.7) jsou parametry, které upravují zvýraznění hran.

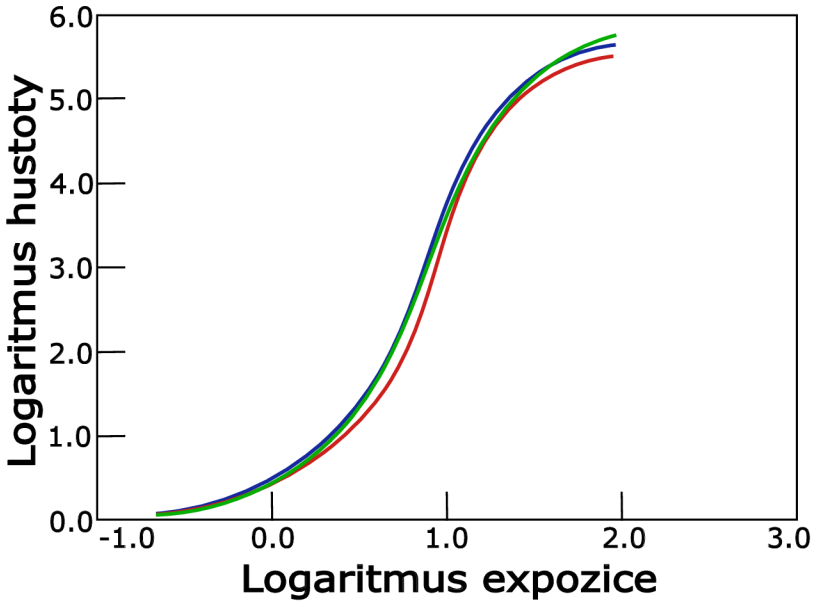
Globální operátor poskytuje tedy dostatečné výsledky, je velmi rychlý a vhodný pro implementaci v real-time aplikacích. Pokud ovšem scéna obsahuje velmi vysoký rozsah jasů, je pro zachování jemných detailů lepší lokální operátor na úkor výrazného snížení výkonu.

2.5.2 Operátory založené na modelu vzhledu obrazu

Tak se dá přeložit aparát pro mapování tónů tzv. *image color appearance model*. Tento globální TMO je založen na výpočetním modelu zrakové adaptace, který byl upraven tak, aby odpovídal psychofyzikálním výsledkům prahové viditelnosti, barevného vzhledu, zrakové ostrosti a citlivosti v čase.

2.5.3 Operátory filmového mapování tónů

Tento název je doslovným překladem originálního označení *Filmic tone mapping operators*, které vychází z charakteristik světlocitlivých fotografických filmů popisovaných ve filmové radiografii. Tyto charakteristiky se jinak nazývají také „senzitometrické křivky“, „křivky hustoty“ nebo „křivky HD“. Popisují reakci fotografického filmu na dopadající fotony v závislosti na nastavené expozici a lze tím určovat výkon světlocitlivých emulzí filmů. Jak ukazuje obrázek 2.8, senzitometrická křivka je graf vztahu mezi množstvím expozice dodané filmu a jeho odpovídající hustotou po zpracování [8].



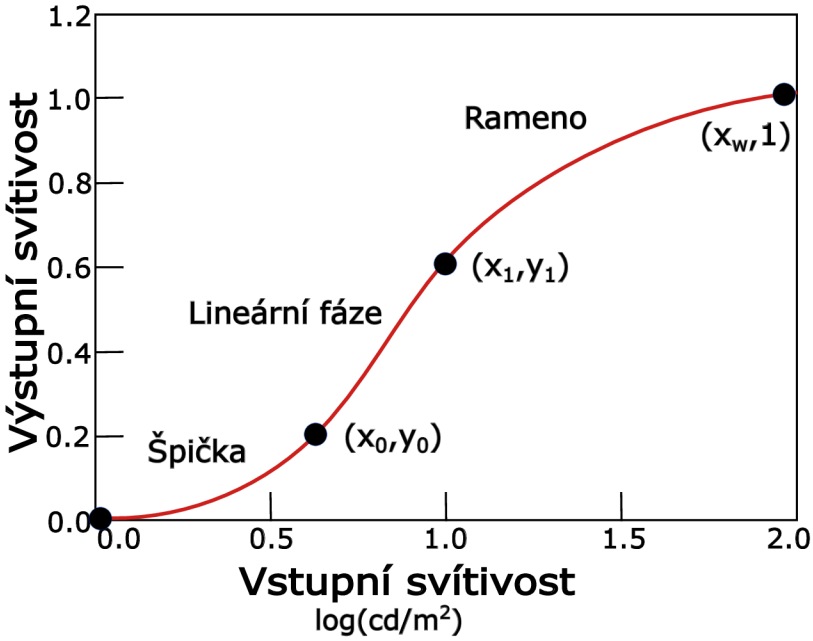
Obrázek 2.8: Křivka hustoty filmu *KODAK VISION Premier color print film 2393*[31]. Hustota zaznamenané barvy na filmu v závislosti na hodnotě expozice. Barva křivky určuje, který z barevných kanálů RGB reprezentuje. Graf je v přibližném měřítku.

[31] Kodak specifikace v souboru *lab_h12393t*: `https://125px.com/docs/motionpicture/kodak/lab`.

Tyto operátory mají za cíl dosáhnout velmi kontrastních obrazů se sytou černou barvou a produkovat vizuálně velmi atraktivní scény, jaké po *post-process* úpravách vytváří ve filmovém průmyslu. Scény mapované těmito operátory evokují dojem filmových snímků a jsou natolik vizuálně přitažlivé, že si je oblíbili vývojáři počítačových her a ve velké míře je implementují ve svých enginech.

Haarm-Pieter Duiker TMO

Jako první aproximoval křivku hustoty Kodak (která je na obrázku 2.8) pro mapování tónů Haarm-Pieter Duiker [11] a poprvé ji uvedl v prezentaci společnosti *Electronic Arts* v roce 2006 jako reakci na příchod HDR dat v herních enginech a next-gen hardwaru. Jeho přístup spočívá ve dvou etapách – transformace lineárních obrazových dat do logaritmického barevného prostoru a následné aplikování LUT[32] na tato logaritmická data. Celý postup je nakonec implementován v HLSL[33] shaderu. Využívá se tvaru „S“ křivky hustoty tak, že střední tóny se lineární fází mapují na největší rozsah jasu, zatímco nejtmavší a nejjasnější hodnoty jsou mapovány na výrazně menší rozsah. Tuto základní podobu přebírají všechny filmové operátory. Liší se podle dané aplikace tím, na jak velké rozsahy jsou jednotlivé fáze křivky nastaveny.



Obrázek 2.9: Mapovací S-křivka s maximální výstupní hodnotou 1.0, což je omezený rozsah LDR. Graf je v přibližném měřítku. Převzato a upraveno [16].

Obecná filmová křivka použitelná pro tónové mapování, jak lze vidět na obrázku 2.9, sestává vždy ze tří částí – špička (tzv. *toe*), lineární fáze a rameno (tzv. *shoulder*). Špička zajišťuje produkci velmi syté černé barvy, střední hodnoty se mapují lineárně a rameno poskytuje jemnější přechod světlých hodnot až do plné bílé.

[32] Vyhledávací tabulka, která v kontextu barev transformuje vstupní hodnoty barev na požadované výstupní hodnoty.

[33] High Level Shader Language (HLSL) – vyšší programovací jazyk pro psaní grafických shaderů.

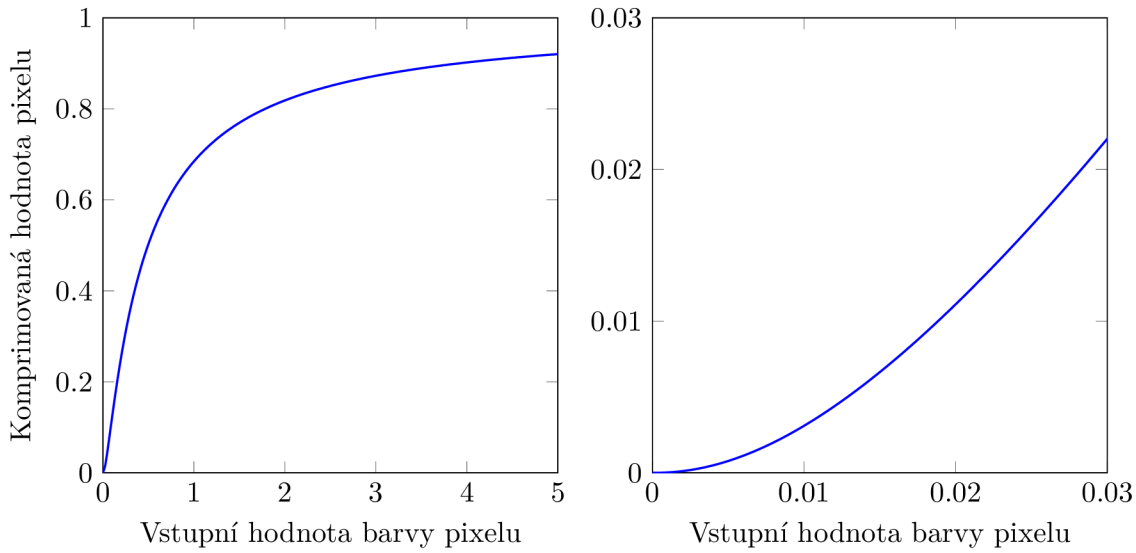
Na práci Duikera později navázali další lidé a myšlenku filmového mapování tónů ještě vylepšili, jak je ukázáno níže.

Hejl-Dawson TMO

Optimalizovanou verzi operátoru od Duikera vytvořili J. Hejl a R. Burgess-Dawson [15]. Podařilo se jim odstranit veškeré převody barev mezi lineárním a logaritmickým prostorem a nutnost využití LUT. Tím výrazně snížili režii, protože využili pouze aritmetiko-logické operace. Jejich funkce, která aproximuje filmovou křivku, je dána jako:

$$y(x) = \left(\frac{x(6.2x + 0.5)}{x(6.2x + 1.7) + 0.06} \right)^g \tag{2.9}$$

kde y je výsledný jas pixelu komprimovaný na LDR rozsah $[0.0, 1.0]$, x je vstupní jas pixelu doplněný o konstanty a exponent g je parametr zabudované implicitní gama korekce. Na obrázku 2.10 je vidět velmi dlouhá lineární fáze a velmi pozvolný přechod k nejjasnějším hodnotám v části ramene. Detail špičky vpravo ve výrazně zmenšeném měřítku ukazuje velmi ostré mapování tmavých pixelů na sytě černé.



Obrázek 2.10: Vlevo průběh Hejl-Dawsonovi mapovací křivky a vpravo detail její špičky, která je bez zvětšení nerozeznatelná.

Uncharted 2 TMO

Výrazným úspěchem v roce 2010 bylo představení globálního filmového TMO pracující na základě filmové S-křivky, který představil vývojář John Hable [15] na GDC[34] a byl poprvé použit ve hře *Uncharted 2*. Stal se velmi populárním a začal se hojně využívat napříč různými interaktivními aplikacemi pro svoji vysokou vizuální atraktivitu.

Hablův přístup spočívá v prodloužení ramene k dosažení velmi pozvolného stoupání k prahové hodnotě maximálního jasu. Čím větší rozsah jasů ve scéně, tím více žádoucí je delší a pomalejší stoupání ramene. Základ operátoru tvoří následující předpis:

[34] Game developers conference (GDC).

$$y(x) = \frac{x(Ax + CB) + DE}{x(Ax + B) + DF} - \frac{E}{F} \tag{2.10}$$

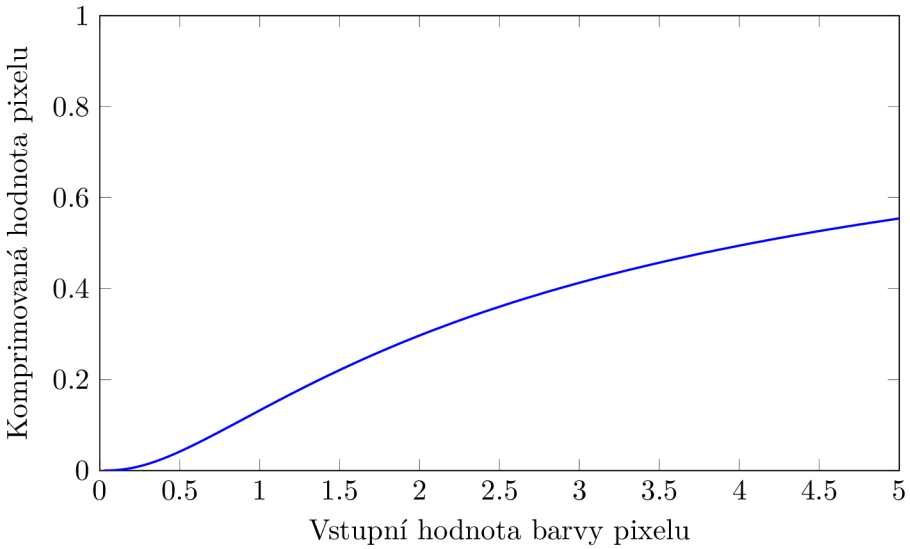
kde x je vstupní hodnota barvy pixelu původního HDR obrazu, y komprimovaná hodnota pixelu a parametry A až F jsou měnitelné koeficienty určující vlastnosti špičky, lineární fáze a ramene. Význam jednotlivých parametrů je následující:

- parametr A = síla ramene, která udává, jak ostrý bude přechod z lineární fáze do části ramene
- parametr B = síla lineární fáze, která udává, jak velký dynamický rozsah bude mapován lineární fází
- parametr C = úhel lineární fáze, který definuje strmost křivky v lineární část
- parametr D = síla špičky, která určuje strmost fáze špičky, tzn. že čím je menší, tím ostřejší je přechod ze špičky do lineární fáze
- parametry E, F = čitatel a jmenovatel špičky, jejich podíl určuje úhel špičky
- parametr W = bílý bod, který se používá ve finálním výpočtu operátoru

Konečná podoba operátoru je dána úpravou bílého bodu následovně:

$$f = y(x)\frac{1}{y(W)} \tag{2.11}$$

kde f je finální komprimovaná barva pixelu v rozsahu LDR, x je vstupní hodnota pixelu původního HDR obrazu a W je parametr bílého bodu. Hable uvádí výchozí hodnoty parametrů takto – $A = 0.22$, $B = 0.30$, $C = 0.10$, $D = 0.20$, $E = 0.01$, $F = 0.30$ a $W = 11.2$.



Obrázek 2.11: Mapovací křivka *Uncharted 2* TMO. Ostrá špička a velmi pozvolně stoupající rameno, které pokrývá široký rozsah jasů.

Jak autor uvádí [16], jednotlivé fáze lze rozdělit pomocí čtyř bodů, což lze vidět na obrázku 2.9. První bod je vždy v počátku a neobsahuje žádné proměnné, protože je jisté, že

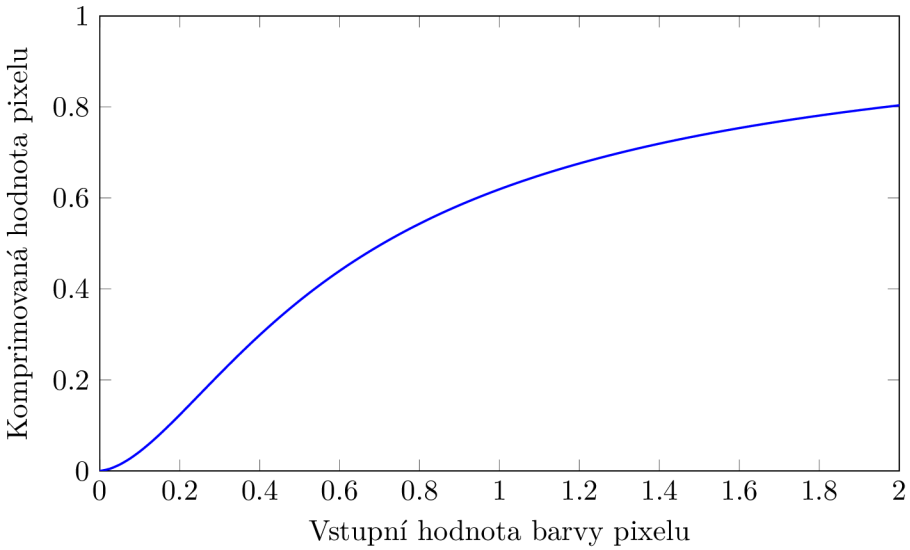
bude vždy v bodě $(0, 0)$. Další dva body vymezují lineární fázi křivky a jsou označeny jako (x_0, y_0) a (x_1, y_1). Čtveřici fázových bodů uzavírá konec ramene, tedy bod $(x_w, 1)$, který je na ose y vždy v bodě 1 (maximální možná hodnota pro LDR zobrazení) a souřadnice x funguje jako parametr – tzv. bílý bod x_w. Pro špičku není potřeba znát parametry, protože je vždy popsána implicitně parametry (x_0, y_0) a sklonem lineární fáze. Po vhodné úpravě lze tyto body následně implementovat tak, že pomocí nich lze snadno upravovat tvar jednotlivých částí špičky, ramene a lineární fáze.

ACES TMO

Academy Color Encoding System [1] je systém kódování barevného obrazu, který zaštiťuje Akademie filmového umění a věd. Projekt ACES vznikl v roce 2004 a je využíván ve filmovém průmyslu pro správu barev při práci s různými typy materiálů, které pocházejí z různých zdrojů, tedy z digitálních kamer nebo tradičních filmových negativů. Systém umožňuje správu rozličných formátů souborů, kódování obrazu, reprodukci barev a další.

V rámci tohoto průmyslového standardu byl vyvinut i globální operátor mapování tónů. V poslední době velmi populární, stal se například výchozím TMO používaným v *Unreal Engine 4*.

ACES definuje kompletní proces práce s barevnými obrazy a v rámci tohoto procesu definuje i transformace *Reference rendering transform* (RRT(x)) a *Output device transform* (ODT(x)), které slouží k převodu barev scény na prostor displeje, což se děje prostřednictvím typické filmové S-křivky.



Obrázek 2.12: Sigmoidní křivka mapování tónů v logaritmickém prostoru ze standardu ACES.

Značně realistický návrh a implementaci od Stephena Hilla publikoval Krzysztof Narkowicz [20]. Na vstupu a výstupu se pomocí transformačních matic provede transformace RRT/ODT a jako mezikrok fitování křivky na tyto transformační data podle předpisu níže:

$$y(x) = \frac{x(x + 0.0245786) - 0.000090537}{x(0.983729x + 0.4329510) + 0.238081} \tag{2.12}$$

kde x je RRT transformovaný pixel původní HDR scény. Výstupní hodnota pixelu $y(x)$ se vynásobí výstupní transformační maticí ODT. Tvar křivky rovnice (2.12) je vynesen do grafu na obrázku 2.12.

Kapitola 3

Návrh demonstrační aplikace

Výstupem této práce je desktopová aplikace demonstrující způsob, jakým je možné renderovat virtuální 3D scény ve vysokém dynamickém rozsahu. Tato kapitola podrobně popisuje základní strukturu celé demonstrační aplikace, z jakých komponent se skládá a na jakém principu celý program pracuje. Jednotlivé implementační detaily jsou následně vysvětleny v další kapitole 4.

3.1 Cíle aplikace

Požadavkem na aplikaci je funkční ukázkové demo, které prezentuje vybrané metody 3D HDR zobrazování. Kapitola 2 je komplexní studie, která se nezaměřuje pouze na jednu úzkou oblast oboru HDR, ale naopak popisuje celou problematiku HDR obrazů, která je velmi rozsáhlá a nezahrnuje pouze problém jasu. Jsou zde obecně i podrobněji popsány veškeré možné techniky, které se v současnosti používají k produkci obrazů ve vysokém dynamickém rozsahu. Po důkladném zkoumání těchto technik a zhodnocení využitelnosti jsem navrhl aplikaci, která využívá metod, které jsou na základě tohoto studia nejdůležitější pro real-time 3D grafiku.

Bez ohledu na účel, který má grafická aplikace s podporou HDR splňovat, je společným požadavkem výkon metod. Z toho důvodu je můj návrh soustředěn co nejvíce na využití akcelerace grafické karty a algoritmy jsou přesunuty do prostoru shaderů.

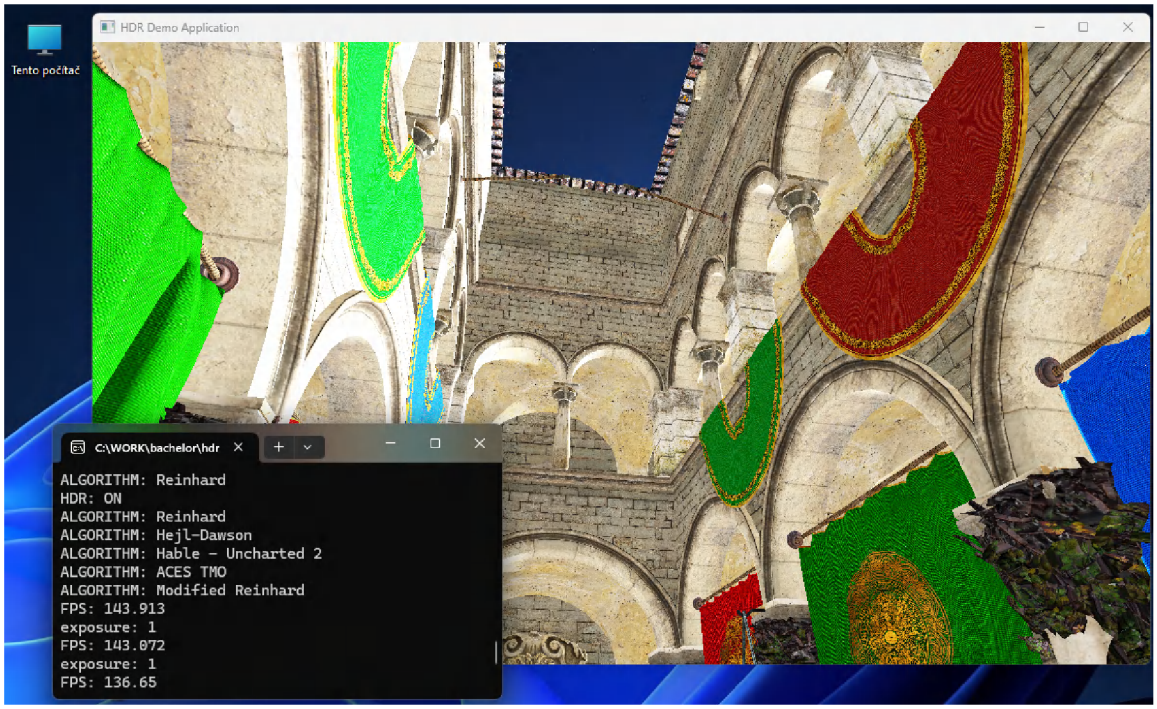
Při návrhu aplikace byla však otázka účelu velmi důležitá. Příkladem odvětví, které je v tomto směru velmi specifické, jsou biomedicínské aplikace. Tam je kladen důraz především na míru věrohodnosti a míru detailnosti, což znamená, že se zde využijí především složité algoritmy, které se snaží při výpočtu zohledňovat co nejvíce okolností a nasbíraných informací. To vede ke značně náročným řešením, které jsou naproti tomu například v oboru počítačových her nebo filmových animací zbytečné. Pokud by měly být stejně složité algoritmy implementovány například v počítačové hře, vedlo by to sice k velmi detailním scénám, ale lidské oko by takový obraz považovalo za nevzhledný a nerealistický, protože by se dívalo na reálná „surová“ data věrně zobrazená na displeji. Naopak mnohem realističtěji vnímá oko scény, ve kterých jsou výrazné kontrastní barvy, kontrastní světla a různé efekty rozmazání okolí zářivých objektů (*bloom* efekt) nebo třeba hloubka ostrosti (*depth of field*). Tyto efekty jsou v počítačových hrách implementovány v rámci tzv. *post-processingu*, na který by při použití zmiňovaných složitých algoritmů z medicínských aplikací ani nezbyl výpočetní výkon a byly by nerealizovatelné.

Proto jsem se v této práci rozhodl navrhnout grafické demo podobné principům počítačových her nebo animací. Pro aplikace tohoto účelu je tedy důležitá nenáročnost a vizuální atraktivnost výsledných snímků. Byly proto vybrány metody především z oblasti mapování tónů a HDR osvětlení.

Zároveň plní navržená aplikace i další smysl. Vzhledem k tomu, že zadání práce klade požadavek na návrh programu v aplikačním rozhraní *Vulkan* a následné zveřejnění pod *open-source* licencí, může tato práce zároveň sloužit jako návod nebo podpůrný zdroj při studiu pokročilejších pasáží tohoto rozhraní. Jelikož je *API Vulkan* poměrně novou technologií, je jeho studium mírně obtížnější s ohledem na množství a dostupnost studijních materiálů. Tato práce je tak v tomto směru jedinečným zdrojem informací a poskytuje základ pro další výzkumné práce nejen v oboru HDR zobrazování.

3.2 Popis výsledného návrhu

Program nese název *HDR Demo App* a jeho cílem je ukázat, jak mohou techniky HDR vypadat ve 3D světě. Program je koncipován jako okenní aplikace (samozřejmě s oknem měnitelné velikosti), ve které ihned po spuštění běží interaktivní virtuální 3D scéna, ve které je možné se s kamerou volně pohybovat („létat“) do všech směrů a rozhlížet se kolem sebe. Výchozí velikost okna po spuštění je *full HD* (1920x1080p). Úspěšně spuštěná aplikace je vidět na obrázku 3.1. Pohyb a ovládání je jednoduché a je zprostředkováno vstupem z klávesnice.



Obrázek 3.1: Snímek obrazovky se spuštěnou aplikací v systému Windows 11.

Scéna obsahuje několik objektů, se kterými však nelze nijak interagovat. Slouží pouze pro vizuální ukázku možností 3D renderování. Kromě objektů tvoří virtuální prostor také *skybox*, tedy forma textury obklopující celou scénu, která tak evokuje dojem, že se objekty a divák nachází v rozsáhlém realistickém světě. Největší objekt tvoří standardní model pro

grafické vizualizace tzv. *Sponza* (historický model atria). Velmi dobře se na tomto modelu prezentují osvětlovací techniky, protože obsahuje venkovní i vnitřní plochy.

Protože k ukázce HDR renderovacích principů je nezbytné aplikování osvětlení na objekty scény, je v demo aplikaci použit realistický model osvětlení tzv. *Phongův model*.

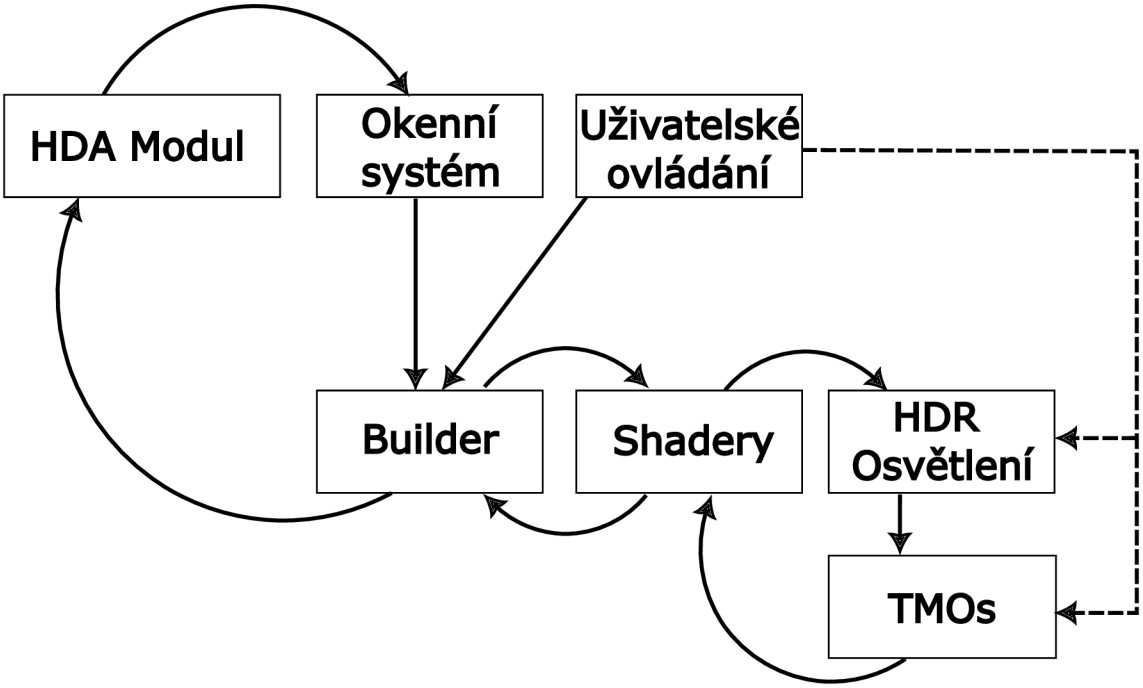
Interaktivita aplikace spočívá v možnosti volného pohybu diváka v prostoru, možnosti spínání světel umístěných ve scéně a ovládání parametrů HDR technik aplikovaných na obraz. Mezi jednotlivými HDR technikami je možné snadno přepínat a vypínat je.

Současně se spuštěním aplikace a otevřením okna s grafickým demem se spustí i konzolové okno. V tomto okně se vypisují různé důležité ladící výpisy, informativní výpisy o uživatelově hardwaru i softwaru, popisy uživatelských akcí provedených v aplikaci a především varovná a chybová hlášení při pádu aplikace.

Ve virtuálním prostředí je možné se pohybovat po neomezenou dobu, dokud uživatel z vlastní vůle program neukončí zavřením celého okna.

3.3 Hlavní komponenty

Obrázek 3.2 představuje kostru programu, jak jsou jednotlivé součásti propojeny a jak mezi sebou komunikují. Implementační detaily budou popsány v další kapitole 4.



Obrázek 3.2: Struktura funkčních bloků aplikace. Směry šipek udávají pořadí provádění. Přerušované šipky značí nepřímé řízení parametrů funkcí v blocích, na které ukazují.

Ústřední částí aplikace je *HDA Modul*, který je vstupním bodem programu a řídí renderování snímků. Před startem samotného renderování provede modul inicializaci nezbytných součástí, jako je nastavení grafické karty podle možností a limitů daného zařízení, inicializaci objektů souvisejících s funkčností vykreslovací pipeline a spuštění komponenty *Okenní systém*. Ten se stará o správu oken, tedy příjem a obsluhu jejich událostí a komunikaci s operačním systémem.

Následně inicializuje i modul *Builder*, který až do ukončení aplikace volá pro vykreslení každého snímku. *Builder* sestavuje jednotlivé součásti vykreslovacího řetězce (*pipeline*), načítá modely a textury a provádí příkazy vykreslování snímků. Tvoří tak logické jádro procesu renderování.

Kromě renderování má *Builder* za úkol také ovládání aplikace, tedy vstup z klávesnice, kterým se řídí pohyb kamery ve scéně, řízení parametrů HDR metod a jejich přepínání. Tuto službu zpracovává komponenta *Uživatelské ovládání*.

Dále se prostřednictvím *Builderu* z grafické pipeline volá blok *Shaderů*, který řídí proces osvětlení a tónového mapování. Tento proces je rozdělen do samostatných funkčních částí, přičemž nejprve se provádí funkce *HDR Osvětlení*, která provádí Phongův osvětlovací model ve vysokém dynamickém rozsahu. Po výpočtu osvětlení scény následuje blok *TMOs* řídící metody HDR pro mapování tónů.

Builder

Jak již bylo zmíněno výše, *Builder* je jádrem vykreslování snímků. Propojuje několik modulů, které se vzájemně doplňují. Před samotným renderováním musí sestavit scénu, k čemuž je potřeba načtení objektů, které scénu tvoří. Ve výchozím nastavení tohoto dema jsou ve scéně dva objekty a skybox. Návrh aplikace dovoluje uživateli vyměnit tyto objekty za vlastní, například z důvodu testování vlastní práce nebo pro analýzu použitých HDR metod na jiných modelech. Stačí do adresáře `\hdr_demo_app\models\` přidat vlastní model souborového formátu *OBJ Wavefront*, který ovšem musí mít název tvaru `m-X.obj`, kde `X` je číslo modelu 1 nebo 2. Textury těchto nových objektů se musí nahradit v adresáři `\hdr_demo_app\models\textures\m-Xtex\`, kde `X` opět značí číslo modelu a musí korespondovat s číslem, které je v názvu souboru daného modelu. Skybox je také možné vyměnit za vlastní. V adresáři `\hdr_demo_app\models\textures\skybox\` nahradit jednotlivé textury za vlastní, opět s dodržením formátu názvů, který je zřejmý z obsahu adresáře `\skybox\`.

Builder kromě načítání objektů také vypisuje většinu informací v konzolovém okně, například hodnotu FPS[1] (jejíž výpočet provádí přímo *Builder*), která může uživateli sloužit jako rychlý odhad okamžitého výkonu aplikace nebo informace z modulu *Uživatelského ovládání*.

Další funkcí *Builderu* je předvýpočet vysokého dynamického rozsahu světelných zdrojů. Hlavní práci, tedy výpočet dopadajících paprsků, jejich odrazy a odlesky se provádí v rámci Phongova osvětlovacího modelu v modulu *HDR Osvětlení*, ale hodnoty vysokých jasů se určují ve funkcích *Builderu*.

Shadery

Nemalá část výpočtů probíhá právě v komponentě *Shaderů* . Podrobnosti o implementaci, tak jako i u ostatních zde popisovaných blocích, jsou rozebrány v kapitole 4. Aplikace využívá zvlášť shadery pro vykreslování skyboxu a zvlášť pro ostatní objekty scény. V rámci těchto *Shaderů* se provádí moduly *HDR Osvětlení* a *TMOs*. *TMOs* obsahuje metody tónového mapování, které jsou přepínatelné mezi sebou a zároveň i vypínatelné, pro možnost vizuálního porovnání kvality a případného ovlivnění výkonu aplikace, což je součást požadavků zadání práce (kompletní testování dokumentuje kapitola 5. Osvětlení bodovými

[1] Frames per second (FPS) – počet snímků za sekundu.

světly lze také vypínat a zapínat. Toto je řízeno z bloku *Uživatelské ovládání* prostřednictvím *Builderu*.

Uživatelské ovládání

Tento blok řídí obsluhu celého programu a ovládání i výpočet pohledu kamery. Obsluha je popsána v další podkapitole. Výpočet projekčních matic na základě vstupu z klávesnice je realizován ve zvláštních funkcích tohoto modulu, který však funguje opět ve spolupráci s *Builderem*.

Rozhlížení s kamerou diváka ve scéně je možné tradičními klávesami `W`, `A`, `S`, `D` ve smyslu nahoru, dolů, doleva a doprava. Toto rozhlížení by bylo intuitivnější prostřednictvím zařízení myši, ale v případě spuštění aplikace například na notebooku by bylo ovládání skrze touchpad nanejvýš nepohodlné. Navíc s ohledem na to, že v programu jde pouze o „létání“, bylo navrženo toto přenositelnější řešení.

Pohyb ve smyslu dopředu a dozadu je realizovaný klávesami `E` a `Q`, které se nachází hned nad klávesami rozhlížení a ovládání je tak poměrně snadné pomocí prstů jedné ruky.

Světelné zdroje, které představují bodová světla rozmístěná po scéně, jsou ve výchozím stavu vypnutá a je možné je zapnout podržení klávesy `O` (od slova osvětlení). Další klávesy jsou popsány v obecné obsluze programu.

3.4 Obsluha aplikace

Program po spuštění ihned zobrazí okno s vykreslenou scénou, ve které je možné se okamžitě pohybovat, měnit parametry HDR algoritmů nebo je přepínat. Pohyb byl popsán v předešlém odstavci. V této podkapitole bude uveden výčet HDR funkcí a kláves, kterými se obsluhují. Pro pohodlnější práci je přehledný textový dokument o řízení celé aplikace přiložen v souborech s programem. Soubor lze nalézt pod názvem `CONTROLS.txt` a nachází se v adresáři `\bin\` se spustitelným souborem aplikace.

Veškeré informace o tom, které funkce jsou právě používány a jaké hodnoty daný parametr nabývá, jsou vypisovány v konzolovém okně a uživatel tak má přehled i o historii svých akcí, jak ukazuje obrázek 3.3.

1. Zapínatelná funkce HDR

   - Tato funkce slouží k přepnutí vykreslené scény do stavu, kdy je obsah tónově mapován z HDR na LDR obraz. Po spuštění je ve výchozím nastavení funkce vypnuta, tzn. že na LDR displeji bude scéna velmi přesvícená a vizuálně nevzhledná. Zapnutí je možné pomocí stisku klávesy `X` a opětovným stisknutím se funkce vypne.

2. Volba metody mapování tónů

   - Aplikace obsahuje celkem 5 různých algoritmů mapování tónů, mezi kterými lze přepínat. Informace o tom, jakou metodou je momentálně obraz na displeji komprimován, je vždy vidět v konzolovém okně. Výchozí metoda po spuštění programu je *Reinhard TMO* a na další se dá přepnout stiskem klávesy `M`. Každým dalším stisknutím se přepne na další algoritmus, až se postupně uživatel vrátí zpět na výchozí metodu. Pořadí algoritmů je následující – *Reinhard TMO*, *Hejl-Dawson TMO*, *Hable-Uncharted 2 TMO*, *ACES TMO*, *Modified Reinhard TMO*.

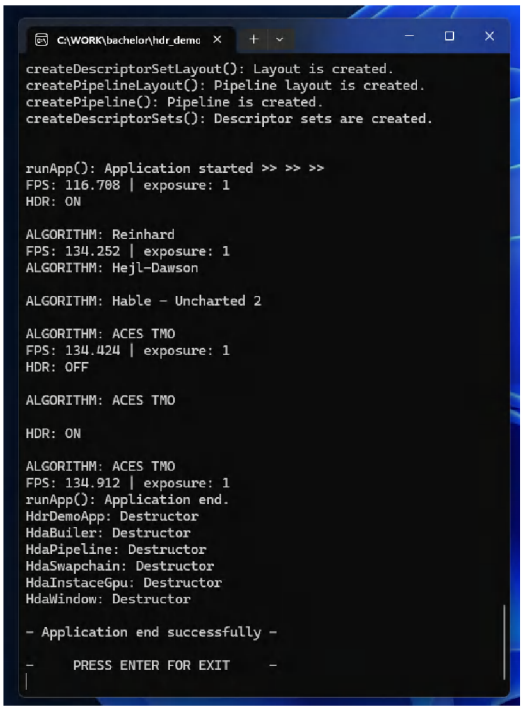
Výběr metody a volba zapnutí funkce *HDR* jsou na sobě nezávislé akce, lze tak snadno pozorovat rychlým přepínáním vizuální (i výkonové) rozdíly mezi jednotlivými metodami.

3. Úprava parametru expozice

   - Expozice je konkrétně v aplikaci navržené touto prací parametr, který udává, zda mají být zvýrazněny detaily ve velmi světlých nebo ve velmi tmavých oblastech (podobně jako u fotoaparátů). Výchozí hodnotou expozice je `1.0`, kterou lze libovolně snižovat i zvyšovat. Různé nastavení produkuje u každé metody různě kvalitní výsledky. Aby bylo možné testování extrémních případů, není změna hodnoty nijak omezena. Nesmyslně velké či malé hodnoty však nemá význam nastavovat, protože vedou k nesmyslným vizuálním výsledkům algoritmů. Zvyšování i snižování expozice se děje skokově po krocích `0.008`. Jelikož je to poměrně malé číslo, budí úprava expozice dojem velmi plynulé změny. Zvyšování se provádí podržením klávesy `C` a snižování podržením `Y`.

4. Ukončení aplikace

   - K ukončení aplikace stačí pouze zavřít hlavní okno a poté pro úplné opuštění potvrdit stiskem `enteru`. Potvrzení se vyžaduje proto, aby měl uživatel k dispozici historii výpisů v konzolovém okně. K ukončení programu může také dojít v případě výskytu chyby. Program může vracet buď pouhá upozornění s příznakem `[WARNING]`, která nevedou přímo k ukončení aplikace nebo chyby s příznakem `[ERROR]`, které okamžitě vedou ke konci programu a čeká se opět na potvrzení klávesou `enter`.



Obrázek 3.3: Konzolové okno s historií výpisů po úspěšném ukončení programu.

Důvodem pro návrh obsluhy pouze pomocí klávesnice je jednoduchost ovládání a přenositelnost. Pro tento typ aplikace, která neobsahuje příliš mnoho nastavitelných parametrů je to naprosto dostačující řešení. Navíc je výhodou tohoto provedení fakt, že grafické uživatelské rozhraní nezasahuje do obrazu scény a je možné tak využít plného rozlišení obrazovky, bez rušivých prvků *GUI*, protože konzolové okno lze vždy zobrazit jen když to uživatel potřebuje a zase jednoduše skrýt. Další výhodou je historie výpisů, která v konzolovém okně setrvává až do zavření a uživatel má přehled o tom, jaké akce provedl. Po případném rozšíření aplikace a přidání nových nastavitelných parametrů by grafická nástavba byla vhodnější.

3.5 Omezení aplikace

Aplikace je limitována určitými omezeními, především implementace stínů a *post-processing* efektů. Vzhledem k náročnosti těchto témat bylo nakonec od jejich aplikování upuštěno, jelikož se v obou případech jedná o techniky, které si žádají adekvátní prostor v samostatné práci. *Post-processingové* efekty k oboru renderování v HDR sice patří, podporují a zkvalitňují jeho výsledky, ale nejsou dle mého názoru nedílnou součástí, bez které HDR renderování nemůže fungovat.

Kapitola 4

Implementace

V této sekci je návrh programu uvedený v předešlé kapitole rozebrán na konkrétní části a podrobně popsán po technické implementační stránce.

Programovým výstupem této práce je okenní desktopová aplikace zobrazující 3D grafiku v reálném čase. Aplikace byla vyvíjena pod operačním systémem *Windows 11 64-bit* ve vývojovém prostředí *Microsoft Visual Studio 17 2022*. Dalším podporovaným operačním systémem je *Windows 10*.

Aplikace byla napsána v programovacím jazyce `C++17`. Jelikož tento jazyk nabízí možnost objektově orientovaného návrhu, bylo této skutečnosti využito a program je konstruován pomocí tříd.

Ze zadání bylo hlavním požadavkem na implementační řešení využití rozhraní pro programování aplikací *API Vulkan*[1]. Jedná se o zcela novou technologii, která byla poprvé vydána v roce 2016. Oproti ostatním tradičním API[2] pro počítačovou grafiku, jako je například *OpenGL*[3], je Vulkan naprosto odlišný svým nízkoúrovňovým přístupem. V této práci bylo využito několik studijních zdrojů, které silně doporučuji pro začínající i pokročilé projekty Vulkan. Předně oficiální dokumentace *Khronos group* [14], tištěné publikace [17] a výukové seriály dostupné online [22, 4, 23]. Další užitečné zdroje v poznámce pod čarou[4].

Dalšími knihovnami použitými při vývoji jsou matematická knihovna `GLM`[5] pro výpočty s maticemi, tzv. *single-file* knihovna `stb_image`[6] pro snadnější načítání textur, další *single-file* knihovna `tinyobjloader`[7] sloužící k pohodlnějšímu načítání objektů formátu `OBJ` a konečně multiplatformní knihovna `GLFW`[8] pro okenní správu.

Následuje popis jednotlivých součástí, ze kterých se skládá kostra aplikace.

Vulkan-hpp

Před samotným popisem funkcí Vulkan je důležité zmínit, že Vulkan API funguje na bázi jazyka C. Společnost Khronos však poskytuje hlavičkové vazby přes tzv. *Vulkan-Hpp* a tato bakalářská práce této vazby využívá. Umožňuje to použití STL kontejnerů, bitových polí,

[1] Oficiální webové stránky zde: `https://www.vulkan.org`.
[2] Application programming interface (API) – rozhraní pro programování aplikací.
[3] `https://www.opengl.org`.
[4] `https://kohiengine.com`
[5] `https://glm.g-truc.net/0.9.9/`
[6] `https://github.com/nothings/stb`
[7] `https://github.com/tinyobjloader/tinyobjloader`
[8] `https://www.glfw.org`

použití výjimek pro ošetřování chyb, jednodušší práci s Vulkan strukturami a další funkce[9]. Definuje také jmenný prostor `vk::` pro třídy a metody.

4.1 Inicializace objektů Vulkan

Prvním a nejzásadnějším bodem programu je inicializace objektů rozhraní Vulkan, což je velmi specifický proces a oproti tradičním přístupům například OpenGL je mnohem složitější. Popsány budou základní objekty, které Vulkan vyžaduje pro správnou činnost. Na konci této podkapitoly je uveden nákres, jak jsou tyto elementy logicky propojeny. Jedná se o obrázek 4.1, který zobrazuje obecnou strukturu Vulkan rozhraní, která je implementována v mojí aplikaci.

V tomto textu bude pomocí jmenných prostorů odlišováno, které metody jsou navrženy a implementovány mnou, a které metody patří rozhraní Vulkan. Všechny metody patřící třídám začínající zkratkou `Hda` (například `HdaInstanceGpu`) jsou dílem mé implementace. Všechny metody začínající zkratkou `vk` patří knihovně Vulkan. Pokud není uveden žádný jmenný prostor, je rozlišení uvedeno v textu.

Instance

Celý proces vytvoření instance, fyzického a logického zařízení je implementován třídou `HdaInstanceGpu` a spouští se z *HDA Modulu* jako první (viz obrázek 3.2). Nejzákladnějším nastavením je pro správnou funkci Vulkan rozhraní vytvoření *instance* (toto označení bude dále v textu znamenat vždy *Vulkan instanci*), což je objekt sloužící k propojení aplikace a knihovny Vulkan. Je to první konfrontace aplikace s uživatelským systémem, pokud funkce `vk::createInstance()` skončí neúspěšně (čili zařízení nepodporuje Vulkan loader[10]), vyvolá výjimku, kterou aplikace obslouží ve funkci `main()`.

Toto je výhoda Vulkan-Hpp proti standardnímu C API, ve kterém se musí kontrolovat návratové kódy knihovních funkcí. Před vytvořením instance je třeba zkontrolovat, zda je dostupné rozšíření pro `GLFW` pomocí `HdaInstanceGpu::checkExtensionSupport()`.

Fyzické zařízení

Dalším důležitým krokem k tomu, aby bylo možné vykreslovat objekty pomocí rozhraní Vulkan, je výběr a inicializace tzv. fyzického zařízení, což je označení pro grafickou jednotku. Její výběr provádí funkce `HdaInstanceGpu::findPhysDevice()`, která nejprve skrze metodu instance získá vektor všech dostupných podporovaných grafických procesorů na daném zařízení. Pokud je seznam prázdný, znamená to vyvolání výjimky a ukončení aplikace, protože nemá smysl dále pokračovat.

Tento výběr je navržen velmi specificky. Funkce vybere pomocí systému hodnocení tu grafickou jednotku, která má nejvyšší rozsah hodnot barevných kanálů pro *surface*. Surface je objekt abstrahující prostor displeje, do kterého lze renderovat grafický obsah a čím více bitů pro uložení barev, tím kvalitnější bude výsledný vyrenderovaný obraz. Protože aplikace je zaměřena na HDR vykreslování, je jako nejpreferovanější formát vybrán `eR16G16B16A16Sfloat` – formát v pohyblivé řádové čárce s poloviční přesností. Tento formát jsem vybral proto, že 32-bitová přesnost by znamenala až přehnaně velkou režii ukládaných dat a 16-bitů je tedy dostačujících. Pokud tento formát grafická jednotka (dále

[9] `https://github.com/KhronosGroup/Vulkan-Hpp`
[10] `https://github.com/KhronosGroup/Vulkan-Loader`

jen GPU) nepodporuje, je jako druhý formát vybrán 10-bitový `eA2B10G10R10UnormPack32`. Pokud GPU nepodporuje ani jeden, je nastaven standardní 8-bitový `eB8G8R8A8Srgb` formát s alfa kanálem a jako barevný prostor je vybrán nelineární `eSrgbNonlinear`. V době psaní této práce jsou aplikací podporovány pouze nejnovější Win 11 a Win 10. Tento minimální formát by tak měl být dostupný na všech GPU obsažených v zařízeních s těmito operačními systémy.

Při výběru GPU je dalším důležitým aspektem to, zda podporuje rozšíření s označením `VK_KHR_swapchain`, bez kterého nelze vykreslovat do obrázků, které *swapchain* spravuje a poskytuje k zobrazení na obrazovce. Krom tohoto rozšíření se selekce grafických jednotek dále řídí i podle toho, zda disponují grafickými frontami (pro příkazy vykreslování) a prezentačními frontami (pro příkazy vykreslování na obrazovku, tedy na surface okna).

Požadavky na typ grafické karty, tj. zda má být dedikovaná nebo integrovaná, aplikace nevyžaduje.

Logické zařízení

Logické zařízení je objekt Vulkanu, který je ústředním objektem v procesu vykreslování. Vytváří se z fyzického zařízení a funguje jako jeho abstrakce, tedy rozhraní, prostřednictvím kterého může aplikace s fyzickým zařízením komunikovat, posílat mu příkazy, přistupovat do paměti a podobně.

Po úspěšném výběru vhodného fyzického zařízení se logické zařízení vytvoří pomocí metody `HdaInstanceGpu::deviceInit()`. Celá třída `HdaInstanceGpu` abstrahuje toto logické zařízení jako jeden velký komplexní objekt. Jednotlivé Vulkan objekty jako instance, zařízení, fronty a další uchovává jako své privátní atributy a poskytuje je bezpečně pomocí metod typu `get()`. Od chvíle vytvoření tohoto logického zařízení se všude dále v programu komunikuje s GPU výhradně prostřednictvím objektu této třídy `HdaInstanceGpu`.

Swapchain

Swapchain je ve Vulkanu objekt, který obsahuje buffery (`vk::Image`), do kterých se vykresluje obsah a tyto buffery (jinak řečeno obrázky nebo snímky) jsou pak zobrazovány na obrazovku. Toto zobrazování na displeji je synchronizováno způsobem, který určuje nastavení swapchainu a podle možností obrazovky, tedy podle obnovovací frekvence.

V aplikaci je swapchain reprezentován třídou `HdaSwapchain`. Nejprve se v rámci konstruktoru vytvoří atribut `vk::SwapchainKHR`, který má mnoho nastavitelných parametrů, například počet vlastněných obrázků, formát surface, barevný prostor surface a podobně.

Zajímavým parametrem je `vk::PresentModeKHR`, který udává jak mají být snímky na obrazovce synchronizovány. Aplikace vytvořená v této práci používá tzv. *double-buffering*, čili do *back-bufferu* se renderuje obsah a *front-buffer* je paralelně vykreslován přímo na obrazovku. Parametr nastavený na `eImmediate` přepne snímek, který je právě vykreslen do back-bufferu, okamžitě k prezentaci na obrazovku a nečeká na dokončení prezentace aktuálního snímku ve front-bufferu. To samozřejmě není úplně ideální a může to způsobovat trhání obrazu tzv. *screen tearing*. Další možné nastavení, které je zároveň i použito v aplikaci této práce, je `eFifoRelaxed` (pokud není dostupné, vybere se první prvek, který vrátí funkce `vk::getSurfacePresentModesKHR()`. Toto nastavení je kombinací dvou možných módů. V případě, že GPU stíhá vykreslovat snímky dopředu a back-buffer vždy při přepnutí obsahuje celý snímek, chová se swapchain jako v módu `eFifo`. Pokud ovšem snímková frekvence GPU klesne pod frekvenci monitoru, začne se přepínaní chovat jako v módu

`eImmediate` a snímky posílat k prezentaci na obrazovku okamžitě. Velmi kvalitní vysvětlení nejen prezentačních módů obsahuje seriál pana Pečivy [23].

Po vytvoření `vk::SwapchainKHR` se pro tuto strukturu vytvoří `vk::Image`, pro který je potřeba vytvořit i abstraktní rozhraní `vk::ImageView`. Závěrem je nutné pomocí funkce `HdaSwapchain::createFramebuffers()` ještě vytvořit vektor struktur `vk::FrameBuffer`. Struktura framebufferu uchovává objekty `vk::ImageView`. Tento framebuffer je používán objektem `vk::RenderPass` při renderovacím průchodu, jak ukazuje obrázek 4.1.

Metoda `HdaSwapchain::createSwapchain()` je navržena tak, aby bylo možné volat ji opakovaně během renderování snímků při požadavku na změnu velikosti okna. V případě, že uživatel okno roztáhne, zmenší nebo minimalizuje je nutné vytvořit celý swapchain se všemi ostatními strukturami znovu pro novou velikost okna. Požadavek na změnu velikosti je z hlavní renderovací smyčky obsluhován metodou `HdaBuilder::recreateSwapchain()` a je možné jej detekovat dvěma různými způsoby:

- prostřednictvím funkce `GLFW` knihovny `glfwSetFramebufferSizeCallback()`, která vyšle požadavek z třídy `HdaWindow` (ta reprezentuje *Okenní systém* z obrázku 3.2) a v hlavní renderovací smyčce `HdaBuilder::render()` se tento požadavek zachytí a obslouží.
- prostřednictvím návratových hodnot (`vk::Result`) knihovních funkcí používaných v hlavní renderovací smyčce `acquireNextImageKHR()` a `presentKHR()`. Obě tyto funkce mohou vracet kódy `eSuboptimalKHR` a `eErrorOutOfDateKHR`. Tyto návratové kódy znamenají ve skutečnosti chyby, které mohou být vyvolané změnou velikosti okna.

Pipeline

Tento objekt reprezentuje třída `HdaPipeline` obsahující metody pro vytvoření struktury `vk::Pipeline`, struktury jejího rozložení `vk::PipelineLayout` a dalších jejich součástí, například moduly shaderů. Knihovní funkce pro vytvoření objektu `vk::Pipeline` je opravdu rozsáhlá, což dokazuje počet osmnácti parametrů, které vyžaduje pro své nastavení. Parametry popisují všechny etapy, ze kterých se celý průchod skrze pipeline skládá. Od načtení modulů shaderů, vertex shader, přes teselaci a oříznutí *viewportu* až po fragment shader.

Pipeline přijímá kód shaderů prostřednictvím `vk::ShaderModule`. Tyto moduly vytváří konstruktor třídy z `SPIR-V`[11] kódu, do kterého je nutné shadery z jazyka GLSL[12] zkompilovat před vlastním sestavením celého programu. SPIR-V je mezijazyk, který disponuje výbornou přenositelností. Moduly pro všechny shadery používané v programu se vytváří metodou `HdaPipeline::createShaderModules()`.

Command buffery

Jedná se o speciální buffery, které Vulkan definuje jako příkazové, tj. ukládají se do nich příkazy vykreslování pro GPU. Tyto příkazové buffery sdružuje objekt *command pool*.

Implementovaná aplikace je navržena tak, že vytváří počet command poolů podle počtu bufferů, které jsou používány při renderování na obrazovku. Navržená aplikace využívá double-bufferingu, tzn. že počet command poolů je 2. Hodnotu double-bufferingu je reprezentována makrem `PARALLEL_FRAMES`.

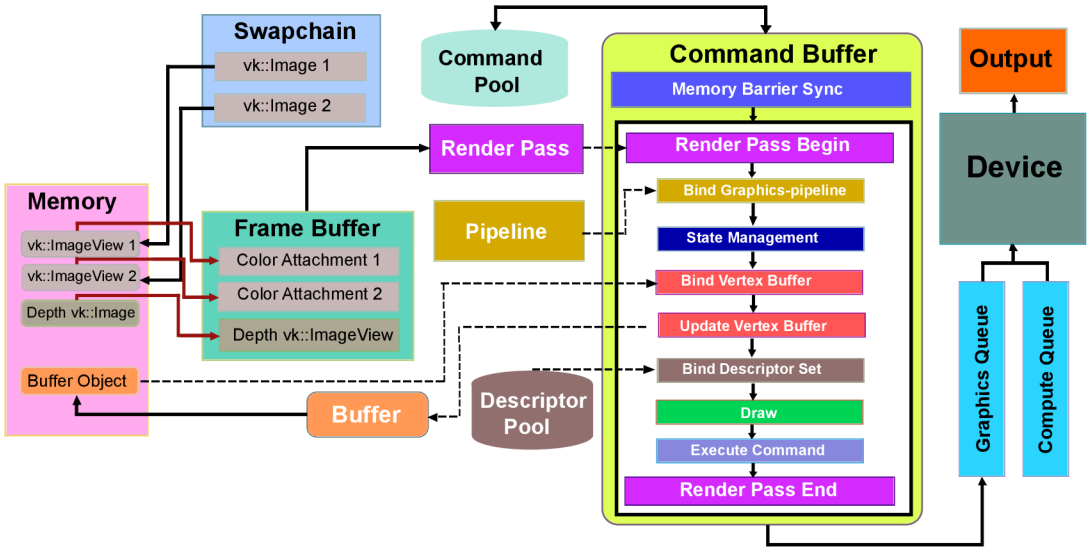
[11] `https://www.khronos.org/spir/`

[12] `https://www.khronos.org/opengl/wiki/OpenGL_Shading_Language`.

Jeden command pool může obsahovat více než jeden command buffer, avšak v navržené aplikaci je v každém command poolu jeden, tzn. jeden command pool a command buffer pro každý z dvojice snímků. O vytvoření se starají `HdaBuilder::createCommandPool()` a `HdaBuilder::createCommmandBuffer()`. Bližší popis práce těchto struktur Vulkanu je v podkapitole o renderování 4.4.

Třída HdaBuilder

Tato třída je jádrem vykreslování. Všechny výše jmenované objekty Vulkanu propojuje dohromady – proto název Builder. Implementuje metody, které připravují všechny další součásti potřebné pro renderování, například načítání a správa modelů, sestavení scény, načítání textur a jejich přiřazení k modelům, vytváření uniformních bufferů, Vulkan deskriptorů, vytváření synchronizačních prostředků pro renderování a další. Ústřední metodou renderování je `HdaBuilder::render()`, kterou volá ve smyčce hlavní *HDA Modul* (viz obrázek 3.2) realizovaný třídou `HdrDemoApp`. Detailní vysvětlení procesu renderování je v podkapitole 4.4.



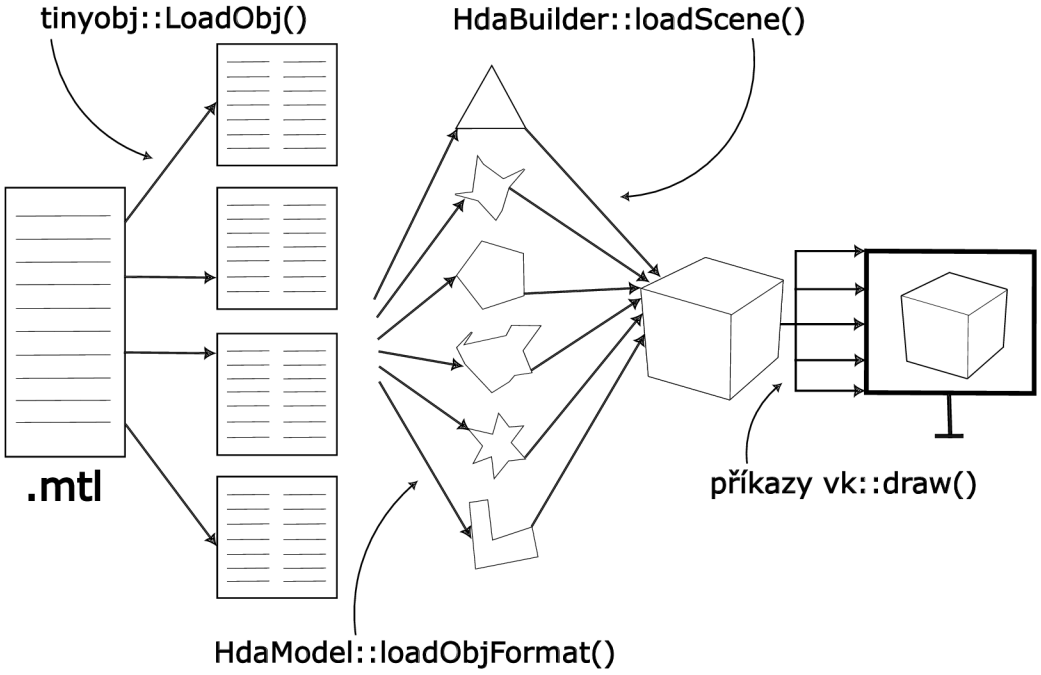
Obrázek 4.1: Struktura procesu vykreslování ve Vulkan API. Anglické názvy jednotlivých bloků jsou nepřeložitelná jména daných objektů knihovny Vulkan. Stejnou architekturu v základu využívá i navržená aplikace. Převzato a upraveno [26].

4.2 Načítání modelů

Pro modely, které se mají načíst a zobrazit ve scéně, je implementována struktura s názvem `HdaModel::SceneObject`, obsahující několik parametrů definujících jeho vlastnosti. Například `HdaModel::Mesh` představující uložené parsované vertexy, vektor struktur typu `HdaModel::Texture` pro texturu modelu, nastavitelné příznaky určující zda má být model vůbec texturován, zda má ve scéně rotovat a nebo například příznak `multiTextureFlag`. Ten rozhoduje, zda se model skládá z několika materiálů a několika různých textur.

Navržená aplikace akceptuje modely ve formátu `OBJ`. Samotné načítání ze souboru abstrahuje metoda `HdaBuilder::loadMesh()`, která ve svém těle volá další funkci s názvem `HdaModel::Mesh::loadObjFormat()` využívající funkcí externí knihovny `tinyobjloader`.

Aplikace v aktuální verzi podporuje pouze dva objekty, které však uživatel může nahradit za vlastní s dodržením určitých kritérií popsaných v podkapitole o obsluze 3.4. V plánovaném rozšíření by tento počet byl určitě vyšší. Model `m-1.obj` je typ modelu, který obsahuje více než jednu texturu a více než jeden materiál. Model `m-2.obj` používá pouze jednu texturu a jeden materiál. Materiálové soubory `MTL` k těmto modelům jsou nezávislé na implementaci a jejich název závisí na tom, jak ho definuje daný `OBJ` soubor, ke kterému patří. Pokud chce uživatel modely nahradit za vlastní, musí dodržet i typy modelů a správně rozlišit, zda jeho model používá více materiálů nebo ne. Na tomto rozlišení totiž záleží při vykreslování. V případě objektu s více materiály se musí příkazy `vk::draw()` provádět zvlášť pro každou část modelu, která sestává ze společného materiálu, až se jednotlivými příkazy `vk::draw()` vykreslí celý model. Toto rozlišování sice znamená striktně deterministický přístup a není tak možné načítat libovolné modely, ale napomáhá to výkonu při vykreslování. Modely s jedním materiálem se jednoduše vykreslí jedním příkazem bez zbytečné další obsluhy.



Obrázek 4.2: Modely

Obrázek 4.2 popisuje postup jednoduchého materiálového systému. Nejprve knihovna `tinyobjloader` pomocí své funkce parsuje soubory `OBJ` a `MTL`. Tím získá oddělená data vrcholů a materiálů. Funkce třídy `HdaModel` dále roztřídí tyto materiály a seskupí vrcholy stejného materiálu do skupin. Každá skupina obdrží index textury, která dané skupině vertexů náleží. Ve třídě `HdaBuilder` jsou při sestavování scény pak objektu přiřazeny z adresáře správné textury podle daných indexů. Ve funkci `HdaBuilder::render()` se pak provádí selekce objektů na ty, které jsou složeny z více materiálů a které ne. Objekt s více materiály je pak renderován po částech s jedním příkazem `vk::draw()` pro každou část – tedy pro každou skupinu vertexů mající stejnou texturu.

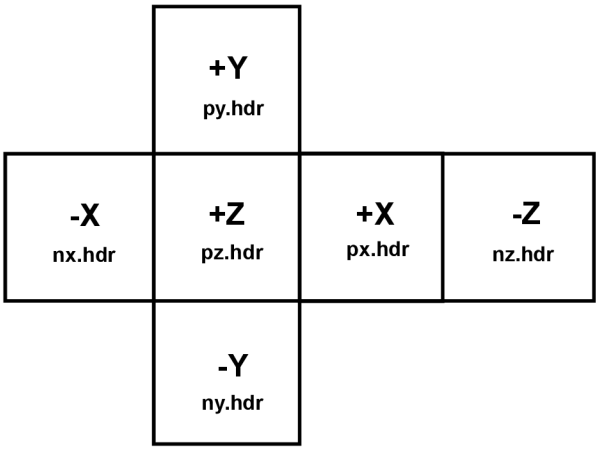
4.3 HDR textury a skybox

Jednou ze základních technik renderování ve vysokém dynamickém rozsahu je použití adekvátních HDR textur. Pokud scéna obsahuje hodnoty osvětlení ve vysokém rozsahu a zároveň textury jsou ve formátu, který nekomprimuje hodnoty barev, může být mapování tónů aplikováno na celý obraz včetně těchto textur a vizuální výsledek je ještě mnohem kvalitnější.

Implementace navržené aplikace toto zohledňuje, avšak s malým omezením. Volně dostupný model formátu `OBJ`, který nabízí svoje textury v HDR formátu, nebyl nalezen. Textury modelů v implementované aplikaci tak využívají pouze 8-bitového RGBA formátu `PNG`. Bude ovšem popsán způsob, jakým lze implementaci snadno upravit pro použití HDR textur.

V případě skyboxu tento problém nebyl a skybox používá texturu ve formátu Radiance RGB (`.hdr`) s rozlišením 8K získanou z webového zdroje volně dostupných HDR fotografií[13].

O načtení textury do struktury `HdaModel::Texture`, která je pak přiřazena k příslušnému modelu, se stará funkce `HdaBuilder::loadTexture()`. Využívá externí knihovny `stb_image` pro extrahování pixelů z textury. Dále se v této funkci vytváří `vk::Image` a `vk::ImageView` pro danou texturu a volá se knihovní funkce `vk::createSampler()`, která vytváří vzorkovač textury, který se používá v shaderech pro mapování souřadnic textury.



Obrázek 4.3: Princip označení jednotlivých částí cubemapy, které používá aplikace. Počáteční písmena v názvech znamenají *positive* a *negative*.

Textura pro skybox má speciální podobu, a je proto řešena ve zvláštní metodě s názvem `HdaBuilder::loadTextureCubemap()`. Skybox je implementován jako krychle větší než celá scéna a všechny objekty se tak nachází uvnitř této krychle. Textura skyboxu, tedy HDR fotografie, musí být rozdělena na 6 stejně velkých čtverců (tzv. *cubemapa*), které jsou naneseny na tuto krychli z vnitřní strany tak, aby složením těchto 6 čtverců vznikla znovu kompletní fotografie. Implementovaným řešením je použití smyčky se šesti iteracemi, kdy se vytvoří jeden `vk::Image` a jednotlivé čtverce cubemapy se nahrají do vrstev tohoto obrázku. Celý obrázek je jako jedna textura následně standardně přiřazen na model krychle. Krychle se nachází jako model `OBJ` v adresáři `\models\` s názvem `sky-cube-def.obj` a textura pro cubemapu je uložena v adresáři `\models\textures\`. Stejně jako modely a jejich textury

[13]Licence Public domain: `https://polyhaven.com`

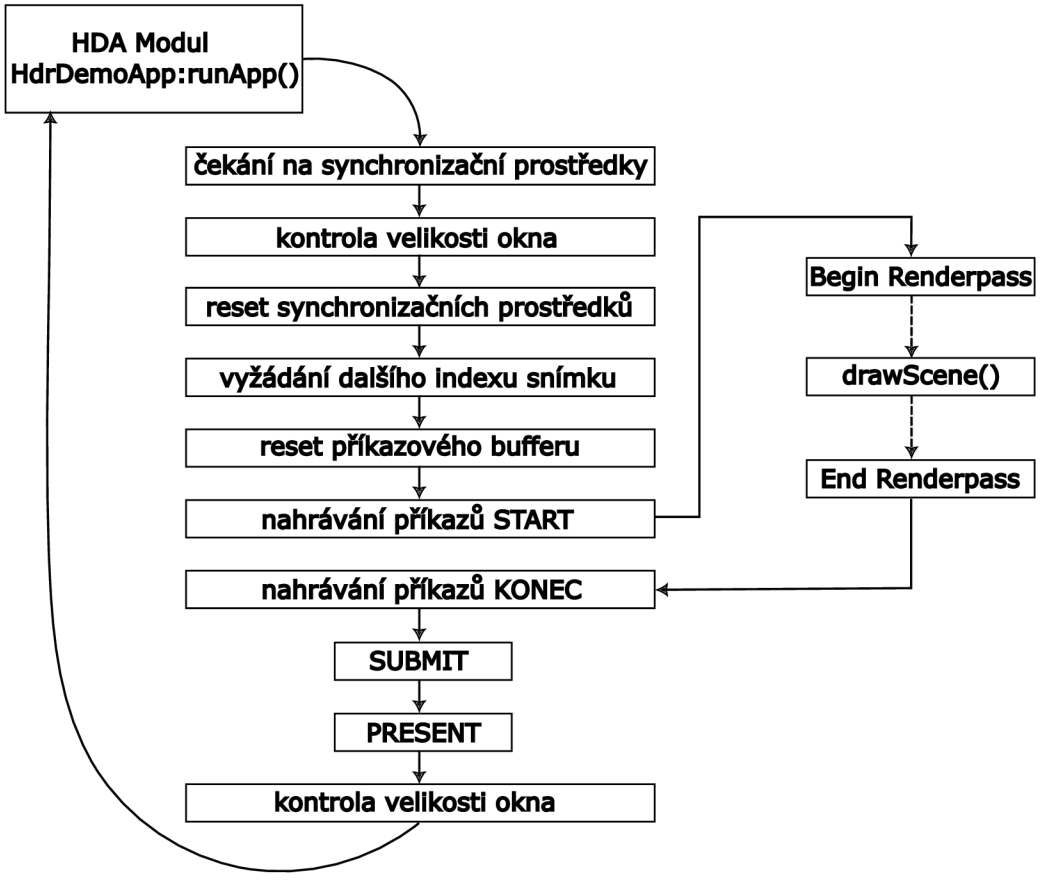
i skybox může uživatel nahradit za vlastní. Musí ale opět dodržet formát názvů, který je vidět na obrázku 4.3.

Důležitým rozdílem mezi metodami načítání textur je použitá funkce z `stb_image`. Pro textury, které mají být aplikovány na objekty, se používá `stbi_load()`, která čte v celočíselném formátu a vytváření objektu `vk::Image` se provádí s parametrem `eR8G8B8A8Srgb` udávající výsledný formát obrázku jako 8-bitový RGBA.

Protože pro skybox se podařilo nalézt volně dostupnou HDR fotografii, je její načítání prováděno funkcí `stbi_loadf()`, která čte čísla v plovoucí řádově čárce. Formát `vk::Image` je při vytváření nastaven na `eR32G32B32A32Sfloat`, čili výsledný skybox je načten ve vysokém dynamickém rozsahu.

4.4 Proces HDR renderování

Hlavní funkce vykonávající proces vykreslování na obrazovku a odesílající příkazy na GPU je implementována v `HdaBuilder::render()`. Základní kroky pro odesílání snímků k renderování a prezentování jsou dané logikou rozhraní Vulkan. Tento postup je zřejmý ze zdrojového kódu funkce `render()` a pro lepší pochopení ještě ilustrován na obrázku 4.4. Celý jeden průchod, ve smyslu šipek od bloku *HDA Modul* a zpět do něj, je paralelně prováděn pro každý snímek. Proměnná `actual_frame` představuje index tohoto snímku.



Obrázek 4.4: Postupné kroky procesu zobrazování rozhraní Vulkan.

Vlastní návrh je použit ve funkci `HdaBuilder::drawScene()`, jejíž největší část tvoří cyklus `for` procházející pole objektů scény. Po navázání pipeline a vyrovnávací paměti vertexů je provedeno větvení podle toho, zda je objekt tvořen jednou nebo více texturami.

Obě funkce se liší pouze ve způsobu zadávání příkazu `vk::draw()` do fronty command bufferu, proto bude uveden popis pouze jedné z nich, a to `drawSingleTexturedObjects()`. Postup práce této metody je následující:

1. Zachytávání stisknutých kláves ovládacích prvků HDR metod čili obsluha aplikace. Děje se tak pomocí callbacků třídy `HdaWindow`. Příznaky, které se při stisku klávesy nastavují, ovládají HDR funkce ve spuštěné aplikaci a jsou do shaderů posílány pomocí deskriptorů.
2. Výpočet projekčních matic v závislosti na pohybu kamery.
3. Nastavení hodnot HDR osvětlení s vysokým rozsahem jasů.
4. Odeslání `push constant` do shaderů.
5. Navázání deskriptorů na dynamické vyrovnávací paměti.
6. Nahrání příkazu `vk::draw()` do příkazové vyrovnávací paměti.

Hlavní přístup této práce spočívá v přesunutí veškerých výpočtů HDR algoritmů na grafickou kartu a aplikování těchto metod okamžitě při průchodu skrze pipeline na každý pixel. To znamená, že všechna práce mapování tónů probíhá ve fragment shaderech. Znamená to obrovskou úsporu výpočetního času oproti očekávanějšímu přístupu v podobě aplikování algoritmů na celkový obraz uložený v back-bufferu a následné poskytnutí obrazu k prezentaci. Čas strávený výpočtem algoritmu celkového obrazu je v real-time aplikacích, jako je ta navržená v této práci, naprosto zásadním problémem.

Oproti tomu řešení pomocí fragment shaderů je obrovská úspora, která se může využít k tomu, aby se ušetřený výpočetní výkon věnoval jiným post-processingovým efektům než mapování tónů. Kvalitní filmové operátory mapování tónů dokáží poskytnout značně kvalitní vizuální výsledek, který podpořený post-processingovým efektem jako je například *bloom*, znamená kvalitní vizuální vjem a zároveň poměrně slušný výkon. Naproti tomu aplikace implementující mapování tónů jako post-processingovou úpravu nemusí najít další výpočetní výkon pro další efekty následného zpracování.

Shadery a TMOs

Program obsahuje dva fragment shadery, jeden pro skybox (`skybox.frag`) a druhý pro ostatní objekty scény (`shader.frag`). Co se týče HDR algoritmů, implementují je oba stejně. Rozdíl spočívá v tom, že `shader.frag` provádí i výpočet Phongova osvětlovacího modelu pro směrová i bodová světla.

Proměnné, které ovládají přepínání mezi TMOs, jsou označeny sifuxem *Flag* a do shaderu se připojují prostřednictvím struktury v dynamické vyrovnávací paměti. Struktura nese název `SceneUniformData` a je definována ve třídě `HdaModel`.

Před samotným použitím některého z TMO je provedeno osvětlení a aplikování textury pomocí `sampleru`. Následuje výběr algoritmu podle hodnoty `chooseMethodFlag`. Výsledek funkce dané metody je vrácen v proměnné `hdrResult`. Pokud není HDR zapnuto, je do výstupní proměnné `outColor` přiřazena hodnota pixelu jako `vec4(result, 1.0)`.

Implementováno je pět algoritmů globálního mapování tónů, z nichž první dostupný po spuštění aplikace je `reinhardTMO(vec3 result, float e)`, kde `result` je vstupní RGB hodnota pixelu a `e` vyjadřuje expozici. Kód rovnice (2.3) je následující:

```
vec3 hdrResult = result / (1.0f + result)
```

Dalším algoritmem je modifikovaný Reinhardův TMO `reinhardModTMO(vec3 result, float e)` využívající exponenciály:

```
vec3 hdrResult = vec3(1.0f) - exp(-result * e)
```

Třetím v pořadí v rámci zdrojového kódu i pořadí ve spuštěné aplikaci je operátor Hejl-Dawson 2.5.3 vytvořený ve funkci `hejlDawsonTMO(vec3 result, float e)` popsaný rovnicí (2.9). Ta je implementována takto:

```
vec3 x = max(((result * e) - 0.004f), 0.0)
vec3 retResult = (x * (6.2f * x + 0.5f)) / (x * (6.2f * x + 1.7f) + 0.06f)
```

Následuje filmová metoda od J. Habla 2.5.3 využívající komplexnější výpočet pomocí rovnice (2.10), která je implementována v `uncharted2TMO(vec3 x)`:

```
vec3 curr = uncharted2TMO(exposureBias * result)
vec3 whiteScale = vec3(1.0f) / uncharted2TMO(vec3(W))
vhdrResult = curr * whiteScale
```

Poslední metodou pro mapování tónů je filmový operátor na bázi specifikace ACES 2.5.3, který nejprve provádí konverzi barevného prostoru transformační maticí `m1`, následně provede vlastní výpočet podle (2.12) a poté je výsledek transformován zpět do sRGB vynásobením `m2`:

```
vec3 v = m1 * result
vec3 a = v * (v + 0.0245786) - 0.000090537
vec3 b = v * (0.983729 * v + 0.4329510) + 0.238081
clamp(m2 * (a / b), 0.0, 1.0)
```

Kapitola 5

Testování a vyhodnocení

Tato sekce se zaměřuje na testování funkčnosti aplikace, srovnání kvalitativních výsledků jednotlivých použitých HDR algoritmů a jejich výkonové srovnání.

Jelikož je obor HDR zobrazování především o vizuální kvalitě, zaměřuje se první část kapitoly na srovnání vzhledu scény různých operátorů mapování tónů. Vizuální dojem je však velmi subjektivní a hodnocení by muselo probíhat ve spolupráci s větším počtem testovacích jedinců, kteří by hodnotili jednotlivé algoritmy nezávisle na sobě. Přesto tato práce využila k posouzení výsledků vyjma autora alespoň dva jedince. Srovnání uvedené v této kapitole tak nabízí kromě porovnání statických obrázků scény také názory jedinců, kteří měli možnost vyzkoušet plný potenciál těchto algoritmů ve 3D real-time pohyblivém světě.

Srovnání vizuální kvality probíhalo na jednom referenčním stroji a ověřování výkonu se provedlo na 4 různých zařízeních, která mají různé výrobce a typy procesorových a grafických jednotek. Všechny zařízení, které budou použity v této kapitole o testování, jsou laptopy. Proto je vynechána tato zmínka o typu zařízení ve všech tabulkách.

5.1 Vizuální srovnání

Pro porovnání byla použita vždy stejná scéna s různou hodnotou expozice pro jednotlivé HDR metody. Vybrán byl takový pohled, ve kterém je vidět zároveň objekt budovy a zároveň i HDR skybox. Testování probíhalo pouze na jednom stroji z toho důvodu, že jeho hardware nabízí širší škálu možností zobrazení obsahu, především počet podporovaných formátu povrchu okna, což umožňuje testovat výsledky s různou kvalitou obrazu.

Označení	Zařízení 1
CPU	AMD Ryzen 5 5600H s Radeon Graphics
GPU	NVIDIA GeForce RTX 3050Ti Laptop
Monitor	Philips 27 palců 2560x1440 144Hz

Tabulka 5.1: Základní HW parametry zařízení.

V tabulce 5.1 je vidět hardwarová výbava referenčního stroje pro vizuální testování. Velkou předností je velký Quad HD displej s 8-bitovou barevnou hloubkou. Jedná se tedy o běžný LDR monitor, ale s poměrně kvalitním kontrastním poměrem 4000:1 a pokrytím barevného gamutu 87.5% AdobeRGB. Zajímavou informací je, že disponuje také funkcí přepnutí do HDR módu se standardem HDR10.

Důvod, proč bylo vybráno toto zařízení je vidět v tabulce 5.2. Nabízí nejvíce formátů výstupního povrchu okna. Scény byly testovány na různá nastavení těchto formátů. Překvapivým zjištěním bylo menší množství podporovaných formátů u GPU od firmy NVIDIA i přesto, že je dedikovaná a patří do vyšší třídy v kategorii grafických karet laptopů. [13]

CPU surface formáty	GPU surface formáty
`B8G8R8A8Unorm`	`B8G8R8A8Unorm`
`B8G8R8A8Srgb`	`B8G8R8A8Srgb`
`A2R10G10B10UnormPack32`	`A2B10G10R10UnormPack32`
`R16G16B16A16Sfloat`	

Tabulka 5.2: Přehled podporovaných výstupních formátů okna pro integrovanou a dedikovanou grafickou jednotku.

Následuje porovnání snímků jednotlivých operátorů. První je uvedena výchozí scéna, která se v této podobě zobrazí po spuštění aplikace. V této scéně není zapnuta funkce HDR a změna hodnoty expozice nemá žádný vliv. Vzadu lze vidět vzdálenou krajinu, což je skybox v souborovém formátu Radiance RGB. Je vidět poměrně sytě bílá obloha a velmi rozmazaná bodová světla kolem domů. V objektu budovy se nachází bodové světlo mířící směrem od kamery kolmo na protější kratší stěnu. Střecha a boky budovy jsou nasvícené směrovým světlem jako simulace slunce. Všechny hodnoty osvětlení jsou počítány v plovoucí řádové čárce a intenzita bodového světla několikanásobně převyšuje rozsah LDR displeje.

Zobrazení této scény na HDR displeji by nemělo způsobovat žádné přesycené bílé plochy. Pokus o zobrazení na LDR displeji však vede ke ztrátě barevné informace. Na dalších obrázcích lze vidět výsledky tónového mapování této scény implementovanými algoritmy s různou hodnotou expozice.

Obrázek 5.1: Výchozí scéna renderovaná s hodnotami osvětlení ve vysokém dynamickém rozsahu zobrazená na LDR displeji bez aplikování operátorů mapování tónů.

Nastavení expozice je pro všechny scény v tomto případě stejné. Testuje se střední hodnota, která by měla vyhovovat všem typům operátorů, tedy 0.159. Formát povrchu okna (surface) je nastaven na `B8G8R8A8Srgb`.

Obrázek 5.2: Reinhard TMO.

Obrázek 5.3: Hejl-Dawson TMO.

Na obrázku 5.2 je výsledek po tónovém mapování nejjednodušším operátorem. Je vidět, že celkové barvy jsou dost ponuré a našedlé. Černé pixely jsou nevýrazné, ale výborně si tento algoritmus poradil s bodovými světly kolem domů na skyboxu. Je vidět, že světla pouličních lamp jsou velmi ostrá a přirozená.

Naproti tomu algoritmus na obrázku 5.3, který je už typem filmových operátorů, poskytuje lehce výraznější kontrastní detaily, lehce sytější barvy v oblasti látek na balkónech a lehce sytější černou barvu v některých malých oblastech.

Obrázek 5.4: Hable (Uncharted 2) TMO.

Obrázek 5.5: ACES (Unreal Engine 4) TMO.

Obrázek 5.4 ukazuje už mnohem lepší výsledek. Oproti Reinhardovi už jsou vidět mnohem větší kontrastní rozdíly v tmavých barvách a velmi dobře si poradil i s krajinou v pozadí. Co má zůstat tmavé opravdu zůstává tmavé, ale pořád tak, aby se daly rozeznat detaily.

Naprosto nejlepší kontrast poskytuje ACES, jeho celkový dojem působí opravdu atraktivně. Velmi dobře vypadá pohled dolů na chodby s barevnými plachtami. Bohužel s tma-

vými místy skyboxu v oblasti lesa selhal. Toto nastavení expozice mu zřejmě nesvědčí pro úplně tmavé místa a pracuje lépe v lehce vyšších hodnotách.

Snímky mapování vybraných metod s nastavením formátu surface `R16G16B16A16Sfloat` lze vidět na následujících obrázcích.

Obrázek 5.6: Nahoře Reinhard TMO a dole ACES TMO.

Výkonnostní testy

Měření výkonu probíhalo na 4 zařízeních s různými procesory a různými grafickými kartami. Průměrná hodnota FPS se měřila po dobu 2 minut pro každou metodu. Během této doby se procházela scéna hlavně uprostřed budovy, kde byla náročnost na vykreslování největší. Náročnost vykreslování skyboxu, a to i v HDR formátu, není na výkonu nijak zna-

telná, o čemž jsem se průběžně přesvědčoval už během vývoje. Testovaná zařízení i s jejich parametry jsou vidět v tabulce 5.3.

Označení	CPU	GPU	OS
Z1	AMD Ryzen 5 5600H 3.3GHz	NVIDIA RTX3050Ti L	Win 11
Z2	INTEL Core i5 7200U 2.5GHz	NVIDIA 940MX L	Win 10
Z3	INTEL Core i3 1115G4 3GHz	INTEL UHD Graphics	Win 10
Z4	AMD Ryzen 5 4600H 3GHz	NVIDIA GTX1650	Win 11

Tabulka 5.3: Seznam testovaných sestav. Záměrně vybrána taková, která se liší výrobcem i typem HW.

Kromě měření průměrné snímkové frekvence pro každou metodu zvlášť se zaznamenávaly i další údaje, například trhání[1] či sekání[2] obrazu. Protože aplikace podporuje pouze OS Windows, jednoduše se využilo při měření jejich funkce *Správce úloh* a zaznamenávalo se využití paměti RAM. Testování se provádělo s 8-bitovým výstupním surface formátem. Výsledky jsou zaznamenané v tabulce 5.3.

Zařízení	FPS	Využití RAM	Tearing x Stuttering
Z1	138	135MB	občasný chvilkový stuttering
Z2	12	200MB	silný stuttering i tearing
Z3	31	650MB	častý stuttering

Tabulka 5.4: Seznam testovaných sestav. Záměrně vybrána taková, která se liší výrobcem i typem HW.

Nejlépe dopadla sestava Z1, ovšem jen v případě, že se v kódu aplikace vyloučil výběr grafické jednotky podle surface formátu. Pokud se kód výběru zařízení vrátil do původní podoby a vybíral opět tu grafickou jednotku, která nabízí nejlepší surface formát, vybrala se na sestavě Z1 integrovaná grafika a výkon klesl na 30 FPS.

Tabulka 5.4 udává průměrné hodnoty pro všechny metody společně, protože se měřením zjistilo, že rozdíl ve výkonu není žádný nebo pouze minimální v případě ACES TMO, kde se všem sestavám lehce propadly FPS o pár jednotek.

Zařízení Z4 ve výsledcích chybí, protože při pokusu o testování selhaly ovladače tohoto zařízení a aplikace vracela chybovou zprávu o tom, že nenalezla žádné fyzické zařízení, které podporuje Vulkan.

5.2 Vyhodnocení

Aplikaci si měli možnost vyzkoušet dva testovací uživatelé. Oba bez předchozích znalostí mapování tónů a technik, které se k tomu používají. Jejich klasifikace jednotlivých metod tak byla zcela objektivní a hodnotili pouze na základě toho, jak se jim obraz líbí a který by chtěli ponechat po zbytek doby strávené v programu. To je v tuto chvíli mnohem lepší metodika hodnocení, než kdybych ji měl provést já, jako zaujatý autor.

Podle očekávání nejlépe hodnotili kvalitu obrazu mapovanou pomocí ACES operátoru. Kladně hodnotili syté černé tóny a celkově měli dojem, že obraz je velmi živý a obsahuje více

[1] Tearing: `https://en.wikipedia.org/wiki/Screen_tearing`
[2] Stuttering: `https://windowsreport.com/fix-pc-stuttering-windows-10/`

barev než ostatní metody. Tímto se dá s úspěšností tvrdit, že nejideálnějším operátorem pro renderování v HDR je ACES TMO. Ovšem záleží pochopitelně na typu aplikace, ve které má být použit. Dle mého názoru by například pro akční hry s nutností dobré viditelnosti v každé oblasti obrazu, ale zároveň s výborným kontrastem a vizuálním dojmem pro lepší požitek z hraní, byl operátor od J. Habla 5.4, který nesaturuje černé barvy až tak výrazně.

Kapitola 6

Závěr

Hlavním cílem této práce bylo ukázat, jakým způsobem je možné tvořit grafické 3D virtuální prostředí, které je vykreslováno ve vysokém dynamickém rozsahu. K tomu, aby mohla být taková 3D scéna vykreslena, je potřeba získat spoustu dalších znalostí z mnoha jiných oborů, ne jen z toho o HDR. Stejně tak tomu bylo i při tvorbě této práce. Před samotným započetím sestavování scény 3D objektů bylo nutné studium širokého spektra témat z počítačové grafiky a především osvojení si základů práce s rozhraním API Vulkan, které by si vzhledem ke svému silnému potenciálu zasloužilo více studijních zdrojů.

Po všech těchto základních znalostech je možné konečně přejít ke studiu samotného HDR, kterému se tato práce věnuje v celé své úvodní polovině. Rozebrány jsou všechny důležité pasáže vysokého dynamického rozsahu, jeho zobrazování i pořizování, zpracování i reprodukce. Tato práce tak slouží jako kvalitní úvod a zasvěcení do problematiky HDR a nabízí v dalších svých kapitolách podrobné aplikování v praxi. Konkrétně v praxi renderování 3D scén, které své osvětlení i povrch textur reprezentují ve vysokém rozsahu jasů, který je však dnes ještě pro drtivou většinu displejů stále nepřekonatelným problém. Na řadu tak musí pořád ještě přicházet již dobře známé operátory mapování tónů, které se však v této práci nesoustředí směrem k oboru fotografie, tak jako mnoho prací předtím, ale konkrétně na aplikování v programech vykreslujících svou grafiku v reálném čase.

Tento přístup uvádím v rámci své práce v jiném úhlu pohledu. Zjednodušit proces převodu HDR snímků na LDR obrazovky a vytěžit z tohoto procesu maximální možnou kvalitu. Toto zjednodušení není zcela nové, ale rozhodně ojedinělé mezi texty takového formátu, jako je bakalářská práce. Výsledkem zkoumání jiných studií, návrhu a implementaci je spustitelná grafická aplikace, která ukazuje, jak lze vykreslovat scény v HDR a navíc formou, která může pomoct nejen odborníkům, ale i nezainteresovaným čtenářům se zájmem o tento obor.

Má práce však stále neobsahuje vše, co by obsahovat mohla. V případě pokračování by jako rozšíření bylo určitě vhodné aplikovat světelné efekty následného zpracování, různé efekty nedokonalosti lidského oka či implementace osvětlení pomocí metody ray-tracing. Jádro aplikace by mohlo být upraveno tak, aby fungovalo jako ladící nástroj pro vývoj nových operátorů mapování tónů. Tzn. vizualizace parametrů operátorů pro jejich snadnější úpravu a přizpůsobení.

Literatura

[1] *Aces* [online]. Dec 2020. Dostupné z: https://www.oscars.org/science-technology/sci-tech-projects/aces.

[2] Adams, A. a Baker, R. *The Ansel Adams Photography Series*. Little, Brown and Company, 1983.

[3] Arbabi, S. *The BETTERPHOTO Guide to Exposure*. Watson-Guptill Books, 2008 [cit. 2023-04-29]. ISBN 978-0817435547.

[4] Blanco, V. [online]. 2020 [cit. 2023-04-10]. Dostupné z: https://vkguide.dev/.

[5] Boitard, R., Pourazad, M. T., Nasiopoulos, P. a Slevinsky, J. Demystifying High-Dynamic-Range Technology: A new evolution in digital media. *IEEE Consumer Electronics Magazine* [online]. 2015, sv. 4, s. 72–86, [cit. 2023-04-29]. DOI: 10.1109/MCE.2015.2463294. Dostupné z: https://ieeexplore.ieee.org/document/7308176.

[6] Bočík, A. *Velká kniha HDR Fotografie: Kouzlo fotografií S Vysokým Dynamickým Rozsahem*. Computer Press, 2009 [cit. 2023-04-29]. ISBN 978-80-251-2098-9.

[7] Chen, H., Zhu, R., Li, M.-C., Lee, S.-L. a Wu, S.-T. Pixel-by-pixel local dimming for high-dynamic-range liquid crystal displays. *Optics express*. Optica Publishing Group. 2017, sv. 25, č. 3, s. 1973–1984. Dostupné z: https://opg.optica.org/oe/fulltext.cfm?uri=oe-25-3-1973&id=357644.

[8] Company, E. K. *Basic photographic sensitometry workbook - kodak* [online]. 2006 [cit. 2023-04-20]. Dostupné z: https://www.kodak.com/content/products-brochures/Film/Basic-Photographic-Sensitometry-Workbook.pdf.

[9] Debevec, P. E. a Malik, J. Recovering High Dynamic Range Radiance Maps from Photographs. In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. USA: ACM Press/Addison-Wesley Publishing Co., 1997, s. 369–378. SIGGRAPH '97. DOI: 10.1145/258734.258884. ISBN 0897918967. Dostupné z: https://doi.org/10.1145/258734.258884.

[10] Drago, F., Myszkowski, K., Annen, T. a Chiba, N. Adaptive logarithmic mapping for displaying high contrast scenes. In: Wiley Online Library. *Computer graphics forum*. 2003, sv. 22, č. 3, s. 419–426.

[11] Duiker, H. P. a Borshukov, G. *Filmic Tonemapping and Color In Games*. Sep 2015. Dostupné z: http://duikerresearch.com/2015/09/filmic-tonemapping-ea-2006/.

[12] Fairchild, M. D. a Johnson, G. M. Meet iCAM: A next-generation color appearance model. In: Citeseer. *Color and imaging conference* [online]. 2002, sv. 2002, č. 1, s. 33–38 [cit. 2023-04-29]. Dostupné z: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=34b7f37f09bcc52897e604dca310f3a0e5fef296.

[13] Ferwerda, J. A. Three varieties of realism in computer graphics. In: Rogowitz, B. E. a Pappas, T. N., ed. *Human Vision and Electronic Imaging VIII*. SPIE, 2003, sv. 5007, s. 290 – 297. DOI: 10.1117/12.473899. Dostupné z: https://doi.org/10.1117/12.473899.

[14] Group, K. *Khronos Vulkan Registry* [online]. [cit. 2023-04-10]. Dostupné z: https://registry.khronos.org/vulkan/.

[15] Hable, J. *Uncharted 2: HDR Lighting* [online]. Naughty dog, 2010. Dostupné z: https://www.gdcvault.com/play/1012459/Uncharted-2-HDR.

[16] Hable, J. *Filmic tonemapping with Piecewise Power curves* [online]. Mar 2017. Dostupné z: http://filmicworlds.com/blog/filmic-tonemapping-with-piecewise-power-curves/.

[17] Lapinski, P. *Vulkan cookbook: Work through recipes to unlock the full potential of the next generation graphics API-vulkan*. Packt, 2017. ISBN 978-1786468154.

[18] Li, Y., Liao, N., Wu, W., Deng, C., Li, Y. et al. Tone Mapping Operator for High Dynamic Range Images Based on Modified iCAM06. *Sensors*. MDPI. 2023, sv. 23, č. 5, s. 2516. Dostupné z: https://www.mdpi.com/1424-8220/23/5/2516.

[19] Mai, Z., Doutre, C., Nasiopoulos, P. a Ward, R. K. Subjective evaluation of tone-mapping methods on 3D images. In: IEEE. *2011 17th International Conference on Digital Signal Processing (DSP)*. 2011, s. 1–6.

[20] Narkowicz, K. *Aces filmic tone mapping curve* [online]. Jan 2016 [cit. 2023-04-20]. Dostupné z: https://knarkowicz.wordpress.com/2016/01/06/aces-filmic-tone-mapping-curve/.

[21] Nightingale, D. *Practical HDR: The Complete Guide to Creating High Dynamic Range images with your digital SLR*. Focal Press, 2012 [cit. 2023-04-29]. ISBN 9780240821221.

[22] Overvoorde, A. *Introduction* [online]. Dostupné z: https://vulkan-tutorial.com/.

[23] Pečiva, J. *Vulkan: Představení a první jednoduchá aplikace.* [online]. Internet Info, s.r.o., Jul 2021 [cit. 2023-04-30]. Dostupné z: https://www.root.cz/clanky/vulkan-predstaveni-a-prvni-jednoducha-aplikace/.

[24] Reinhard, E., Heidrich, W., Debevec, P., Pattanaik, S., Ward, G. et al. *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, 2010.

[25] Reinhard, E., Stark, M., Shirley, P. a Ferwerda, J. Photographic tone reproduction for digital images. In: *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 2002, s. 267–276. Dostupné z:

https://www.researchgate.net/publication/2908938_Photographic_Tone_Reproduction_For_Digital_Images.

[26] Singh, P. *Learning Vulkan: Discover how to build impressive 3D graphics with the next-generation graphics API-vulkan*. Packt, 2016. ISBN 978-1786469809.

[27] Tumblin, J. a Rushmeier, H. Tone reproduction for realistic images. *IEEE Computer Graphics and Applications*. 1993, sv. 13, č. 6, s. 42–48. DOI: 10.1109/38.252554. Dostupné z: https://ieeexplore.ieee.org/document/252554.

[28] Union, I. T. *Recommendation ITU-R BT.2020-2* [online]. 2015 [cit. 2023-04-21]. Dostupné z: https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.2020-2-201510-I!!PDF-E.pdf.

[29] Union, I. T. *Recommendation ITU-R BT.2100-2* [online]. 2018 [cit. 2023-04-21]. Dostupné z: https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.2100-2-201807-I!!PDF-E.pdf.

Příloha A

Obsah přiloženého paměťového média

xrozen02-text-prace.pdf Bakalářská práce ve formátu pdf.

src-prace Adresář se zdrojovými kódy LaTeX.

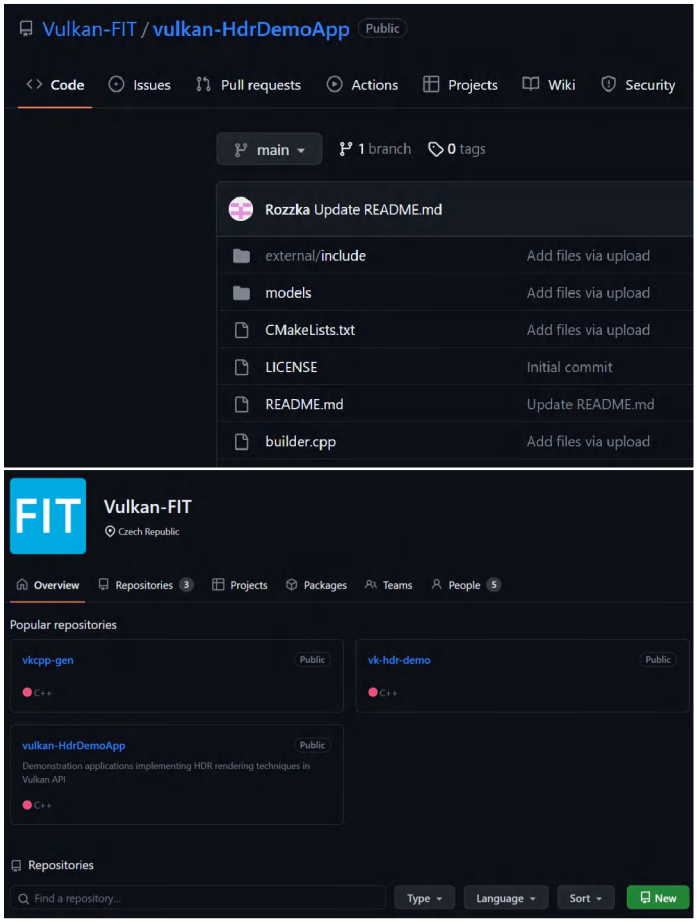
README.txt Soubor readme.

src Adresář se zdrojovými kódy aplikace.

bin Spustitelné soubory aplikace.

Příloha B

Snímek obrazovky zveřejnění zdrojového kódu



Obrázek B.1: Github repozitář zveřejněný pod licencí MIT.