VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

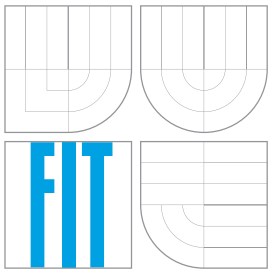
ZPRACOVÁNÍ HDR OBRAZU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE VÍT MOJŽÍŠ
AUTHOR

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZPRACOVÁNÍ HDR OBRAZU

HDR IMAGE PROCESSING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VÍT MOJŽÍŠ

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Dr. Ing. PAVEL ZEMČÍK

BRNO 2014

Abstrakt

Tato práce se zabývá zpracováním HDR obrazu, se zaměřením na návaznost HDR na současné technologie. Praktickým cílem pak byl vývoj uživatelského rozhraní přehrávače HDR videa. Výsledek bude použit pro ovládání nového přehrávače firmy GoHDR. Samotné rozhraní bylo implementováno v prostředí WPF a OpenGL obsah generovaný jádrem přehrávače je zobrazován pomocí SharpGL. Při vývoji byl kladen velký důraz na efektivní používání všech současných funkcí přehrávače a snadnou adaptaci na plánovaná rozšíření.

Abstract

This thesis deals with HDR imaging, focusing on its use with currently available technology. The main aim is to design and build a user interface for controlling HDR video player. Project outcome will be used to operate new player designed by GoHDR. The interface was implemented in WPF, using SharpGL to visualize OpenGL content generated by the player core. Strong emphasis was given to allow effective access to currently implemented player functions, while maintaining enough room for planned extensions.

Klíčová slova

Uživateslské rozhraní, vysoký dynamický rozsah, přehrávač videa, WPF.

Keywords

User interface, high dynamic range, video player, WPF.

Citace

Vít Mojžíš: Zpracování HDR obrazu, bakalářská práce, Brno, FIT VUT v Brně, 2014

Zpracování HDR obrazu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Prof. Dr. Ing. Pavla Zemčíka. Další informace mi poskytli Prof. Alan F. Chalmers, MSc. PhD., Jonathan Hatchett BSc. a Joshua McNamee BSc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Vít Mojžíš
30. července 2014

Poděkování

Rád bych poděkoval profesoru Zemčíkovi za odbornou pomoc. Dále děkuji profesoru Chalmersovi za příležitost, na základě které popisované rozhraní vzniklo, a v neposlední řadě také Jonovi s Joshem za cenné rady a pomoc se zapojením do projektu.

Obsah

Kapitola 1

Úvod

Drtivá většina digitálně zpracovávaného obrazu zdaleka neobsahuje všechny informace, které nabízela její předloha. Tato práce popisuje alternativní přístup k reprezentaci obrazu, jenž lépe odpovídá nárokům reálných scén. Jádrem práce je uživatelské rozhraní přehrávače videa, sloužícímu k zobrazení takto uloženého obrazu na běžně dostupných počítačových monitorech.

Schopnost zaznamenat obraz ve stejné kvalitě, jakou vnímá lidské oko, byla pomyslnou metou technologií pro práci s obrazem pravděpodobně již od pořízení první fotografie. Dalším krokem k tomuto cíli je zavedení metod uchovávajících celý dynamický rozsah scény, známých jako *HDR imaging* (práce s vysokým dynamickým rozsahem). V digitální fotografii je HDR známo už delší dobu díky možnosti sestavit HDR snímek z několika různě exponovaných fotografií pořízených běžným vybavením (*exposure bracketing*). Nasazení ve video průmyslu se začalo objevovat až v posledních letech a je stále většinou ve fázi prototypů.

Vzhledem k nekompatibilitě s aktuálně dostupným zobrazovacím vybavením, je velká pozornost věnována metodám pro snižování dynamického rozsahu HDR obsahu, za účelem zobrazení v běžných podmínkách. Současné metody pro automatizaci tohoto procesu zatím neposkytují ve všech situacích uspokojivé výsledky a jsou proto oblastí aktivního výzkumu.

S touto problematikou jsem se blíže seznámil při zahraniční stáži na univerzitě ve Warwicku, kam jsem se dostal díky spolupráci zmíněné univerzity s ústavem počítačové grafiky VUT v Brně a finanční podpoře programu Erasmus. Náplní stáže byl návrh a implementace uživatelského rozhraní na platformě Windows, sloužícího pro ovládání multiplatformní knihovny LibScarlet, která je navržena pro zobrazování HDR videa mimo HDR prostředí. Jedním z hlavních cílů LibScarlet, je propagace HDR technologií vyvíjených firmou GoHDR, která stojí za jejím vývojem.

Členění práce je následovné. Kapitola 2 vysvětluje základní principy práce s HDR. Kapitola 3 popisuje interakci člověka s počítačem, se zaměřením na uživatelská rozhraní. Kapitola 4 se věnuje existujícím HDR přehrávačům a současným možnostem jejich využití. V kapitole 5 je popsáno vyhodnocování informací z předchozích kapitol spolu s požadavky na rozhraní od firmy GoHDR, a jejich využití při návrhu rozhraní. Kapitola 6 pak obsahuje implementační detaily, průběh testování a možnosti dalšího rozšíření rozhraní.

Kapitola 2

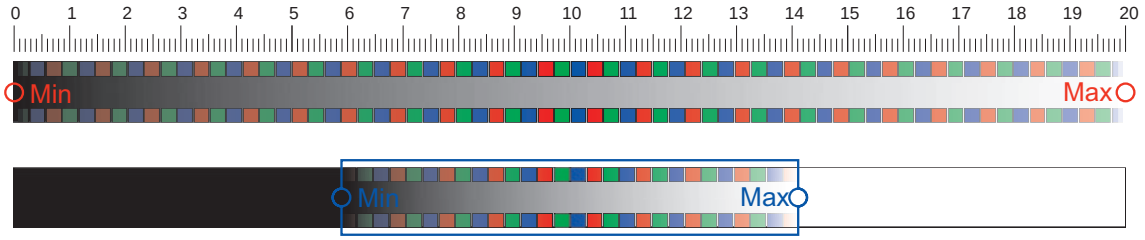
Vysoký dynamický rozsah

Technologie vysokého dynamického rozsahu, neboli HDR z anglického High dynamic range je termín popisující záznam, ukládání, úpravu, přenos a zobrazování obsahu, který přesněji reprezentuje rozsah úrovní světla v reálném světě [2].

Tato kapitola si bere za cíl vysvětlení základních principů HDR, relevantních pro tuto práci, a jejich možného přínosu pro multimédia. Po úvodním nástinu důležitých pojmů používaných ve zbytku práce, následuje zmínka o vztahu lidského vidění a dynamického rozsahu. Závěr kapitoly se pak soustředí na metody adaptace HDR obrazu pro zobrazení na běžně dostupném vybavení.

2.1 Základní pojmy

Reálné scény se liší rozsahem jasů. V některých případech jsou světlá místa mnohonásobně jasnější, než zastíněné oblasti. Pro snazší pochopení proč jsou takové situace problematické pro záznam, zavedeme několik pojmů týkajících se dané problematiky.



Obrázek 2.1: Schéma vztahu mezi dynamickým rozsahem scény (nahoře) a digitálního snímače.

Dynamický rozsah je bezrozměrná veličina, která může být použita k popisu několika různých fyzikálních vlastností. U obrazu jde o poměr mezi nejsvětlejším a nejtmavším bodem, neboli nejvyšší celkový kontrast, který můžeme v daném obraze najít. U displeje se jedná o poměr mezi nejnižší a nejvyšší svítivostí jakou je schopen vydávat. Dynamický rozsah fotoaparátu je poměr mezi osvětlením, které těsně nasytí senzor a osvětlením, které zvedne odezvu senzoru o standardní odchylku nad šumový práh. Dynamický rozsah zapisujeme jako poměr, tedy například 500:1 což znamená, že nejvyšší kontrast v obraze je 500-krát

vyšší, než kontrast nejmenší. Obrázek 2.1[1] ukazuje, co se stane s dynamickým rozsahem scény po vyfocení běžným digitálním fotoaparátem[2]. Senzor fotoaparátu obsáhne většinou pouze část scény a veškeré hodnoty mimo jeho rozsah mají v uložené fotografii maximální, nebo minimální hodnotu. Část detailů z původní scény je tak ztracena. [1].

Scéna, zdroj	**Rozsah jasů**	**Exp. rozsah**
Plný rozsah od Slunce po světlo hvězd	1 000 000 000 000:1	40
Rozsah lidského oka po adaptaci	100 000 000:1	27
Tmavý interiér s výhledem na jasné světlo	5 000-10 000:1	12-14
Lidský zrak v jednom pohledu bez adaptace	10 000:1	13-14
Černobílý negativní film	10 000:1	13-14
Typická DSLR při základní citlivosti	500:1	9
LCD monitor	350:1	8-9
CRT monitor	200:1	7-8
Velmi kvalitní fotopapír	100:1	7
Běžný papír	50:1	5-6

Tabulka 2.1: Porovnání dynamického a expozičního rozsahu několika scén, zobrazovacích zařízení a lidského oka.

Bitová hloubka je počet bitů, ve kterých je uložena kompletní barevná informace jednoho pixelu (BPP – bits per pixel), nebo počet bitů dostupných pro každou barevnou složku (BPC – bits per color/channel) [13]. Kdykoli je tento pojem zmíněn v této práci, vztahuje se k druhé definici (BPC).

Expoziční stupeň, nebo EV (Exposure Value), je absolutní jednotka popisující množství světla dopadající na snímač. Stupnice EV je logaritmická se základem 2. Hodnota 0 EV odpovídá exponování středně šedé tabulky[3] při citlivosti ISO 100, cloně f 1 a času 1 vteřina. Rozdíl mezi nejnižším a nejvyšším expozičním stupněm v dané scéně označujeme jako expoziční rozsah a odpovídá minimálnímu počtu bitů na barevný kanál, potřebných k uložení daného obrazu bez zmenšení dynamického rozsahu. Jedná se o další možný způsob zápisu dynamického rozsahu. Dříve zmíněný zápis získáme dosazením do následujícího vztahu $2^{expozicni\ rozsah} : 1$ [5]. Tabulka 2.1[4] obsahuje několik hodnot dynamického a expozičního rozsahu s příkladem situace, ve které se s nimi můžeme setkat.

2.2 Lidské vnímání dynamického rozsahu

Za normálních podmínek oko vnímá dynamický rozsah asi 1:32 000 (15EV). Adaptací na danou scénu (jednotlivé části oka se dokážou adaptovat nezávisle) je ale schopné dosáhnout rozsahu až 1:1 000 000 000 (30 EV), který se nazývá *absolutní dynamický rozsah.*

Tyto hodnoty však vypovídají pouze o vnímaných extrémech a ne o jemnosti stupnice. Podle takzvaného „Webrova zákona“ závisí minimální rozdíl dvou světelných hodnot, které

[1] Obrázek vytvořen na základě schématu, které uvádí Bočík [3].

[2] Uvažujeme zde dynamický rozsah snímače 8 bitů.

[3] Střední, nebo neutrální šedá, je barva, která vypadá jako střed mezi černou a bílou. Středně šedý povrch odráží 18% dopadajícího světla.

[4] Tabulka převzata z [5].

od sebe oko rozpozná, na velikosti těchto hodnot[5]. Zmíněná závislost je ve většině rozsahu intenzit světla stálá, a pohybuje se kolem 1%. Jinak řečeno, nejjasnější rozlišitelná světelná intenzita ve scéně může být až 10^9-krát jasnější než ta nejslabší, ale abychom rozpoznali rozdíl ve dvou sousedních bodech, musí se v nich intenzita světla lišit alespoň o 1%.

Při reprezentaci obrazu v počítači se proto úrovně světla ukládají nelineárně. Změny v tmavších částech obrazu jsou zaznamenány přesněji než v jasných, čehož je dosaženo gama kompresí. Přibližně lineárně zaznamenané hodnoty z digitálního snímače jsou před uložením komprimovány podle vzorce $Y = X^{1/\gamma}$, kde X je vstup a Y ukládaná hodnota. Hodnota γ závisí na použitém barevném prostoru.

Více o fungování lidského oka a mechanizmech jeho adaptace v kontextu vizualizace uvádí Reinhard a kolektiv [13].

2.3 Možnosti zobrazení

V některých případech je HDR obraz ukládán za účelem dalšího zpracování, jako například identifikace zdrojů světla ve scéně. Nicméně v drtivé většině případů je cílem zobrazení, ať už v tištěné podobně, nebo pomocí zobrazovacího zařízení, například televizní obrazovky, monitoru počítače, nebo projektoru.

Podle doktora Reinharda [13] se technologie pro zobrazení HDR obrazu v blízké budoucnosti stanou dostupnými pro širokou veřejnost a postupně vytlačí většinu současných zobrazovacích technologií. Tento trend potvrzují společnosti SIM2, Sharp, TCL, nebo Vizio, které na HDR displejích pracují. V současné době je ale většina těchto technologií ve fázi prototypů a jejich ceny je činí značně nedostupnými. Kromě toho, tištěná média se pravděpodobně nikdy nestanou HDR[6]. Využití stávajících zobrazovacích technologií (LDR) v souvislosti s HDR obsahem je proto důležité nejen pro úspěšné zavedení HDR, ale i pro následné udržení kompatibility mezi zobrazovacími médii.

Následující část práce se této oblasti věnuje podrobněji. Jsou zde zmíněny základní principy snižování dynamického rozsahu, selektivní zobrazování částí dostupného rozsahu a časté problémy, se kterými se při jejich použití můžeme setkat.

2.3.1 Problémy a omezení

Jak vysvětluje Banterle a kolektiv [1], není možné dostáhnout nerozpoznatelné shody mezi HDR scénou a její reprezentací, pokud monitor neposkytne dostatečný dynamický rozsah a odpovídající barevný gamut[7]. Můžeme ale dosáhnout zachování požadovaných vlastností obrazu, za cenu obětování ostatních. Možnosti dostupných metod pro takové úpravy jsou však omezené, a výsledný obraz je často kompromisem mezi zachováním co nejvíce požadovaných vlastností a omezením negativních dopadů použité metody.

Následuje popis nejčastějších nežádoucích efektů způsobených úpravou dynamického rozsahu obrazu.

Posterizace

Za normálních podmínek je barevný prostor HDR obrazu dostatečný pro zobrazení (pro lidské oko) plynulých přechodů. Výraznější manipulace s poměrem barev v kombinaci s dis-

[5] Weberův zákon platí obecně pro smyslové vjemy, v tomto případě nás však zajímá pouze jeho důsledek na vidění.

[6] Zdůvodnění přesahuje rozsah této práce, ale podrobně se mu věnuje [13].

[7] Barevný gamut v tomto případě definuje množinu barev, které daný monitor dokáže správně zobrazit.

Obrázek 2.2: Ukázka posterizace následkem snížení bitové hloubky.

kretizací barevného prostoru pro LDR displej může vést k viditelným „skokům“ mezi odstíny barev [3]. Pro ilustraci tohoto jevu byla snížena bitová hloubka obrázku 2.2 z 8 na 4 bity.

Halo efekt

Halo artefakty jsou způsobeny mimo jiné lokálním filtrováním dvou sousedních oblastí s velkými světelnými rozdíly. Například pokud je v obraze tmavá plocha blízko okna, kterým prosvítá slunce, jas pixelů v okně se bude propagovat do okolí, což vytvoří světlejší „rámeček“ (halo) [13].

Obrázek 2.3: Ukázka původního obrázku A) a tří variant se sníženým kontrastem. Dvě metody automatického zpracování barev B) a C) ve srovnání s ruční úpravou D).

Zkreslení barev

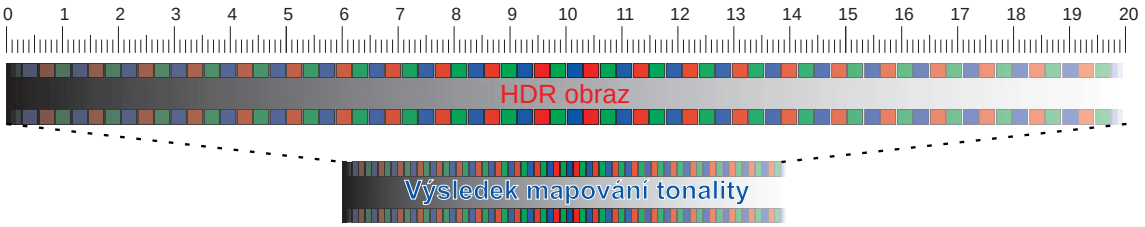
Přesto, že většina mapovacích algoritmů nabízí sofistikované metody pro práci se světelnými hodnotami a jsou schopny jejich věrné reprodukce, vzhled barev ve výsledném obraze je často značně odlišný od původní scény. Na obrázku 2.3[8] vidíme výsledek dvou základních variant automatického zpracování barev, demonstrující nejčastější nedostatek, kterým je špatná saturace barev. Banterle a kolektiv [1] popisuje pokročilejší postupy pro automatickou korekci barev (color appearance models), ale dále uvádí, že jejich výsledky jsou zatím často nekonzistentní a v některých případech je potřebná ruční korekce.

2.3.2 Mapování tonality

Proces snižování rozsahu hodnot v HDR obraze, který umožní následné zobrazení zařízením s nižším dynamickým rozsahem, se nazývá *mapování tonality*. Jednotlivé algoritmy

[8] Obrázek byl převzat z [8].

umožňující tyto úpravy jsou pak známy jako *tonální operátory* (dále jen TMO z anglického Tone-Mapping operators). Většina operátorů se snaží zachovat některé charakteristiky původního obrazu, jako je globální a lokální kontrast, nebo detaily. Cílem je, aby výsledný obraz co nejlépe odpovídal zachycené scéně. Jak demonstruje obrázek 2.4[9], tonální operátory sestavují výsledný obraz na základě informací z celého dostupného dynamického rozsahu [13].



Obrázek 2.4: Schéma komprese dynamického rozsahu při mapování tonality.

Pomyslnou dělící čáru mezi LDR a HDR v tomto případě tvoří rozsah záznamu, který je třeba zobrazit. I pro monitor jako je *Solar47 PRO*, firmy SIM2[10] s 16-ti bitovým rozsahem, bude u některých scén stále vhodné použít metody popsané níže, například při zobrazování fotografií ze *SpheroCam HDR line* firmy SpehronVR[11] s udávaným rozsahem až 26 bitů.

Globální operátory

U globálních operátorů je na všechny pixely vstupního obrazu použitá stejná operace, díky čemuž je zachován globální kontrast. V některých případech se využívá dvou průchodů, kdy v prvním průchodu je obraz analyzován, a výsledky jsou použity k optimalizaci redukce dynamického rozsahu v druhém průchodu. Při analýze je věnována pozornost hodnotám jako minimální a maximální jas, nebo logaritmický a aritmetický průměr všech hodnot.

Kvůli práci s globálními hodnotami, trpí výsledný obraz na změny v lokálním kontrastu a problémy se zachováním jemných detailů.

Lokální operátory

Lokální operátory zvyšují kvalitu výsledného obrazu rekonstrukcí jak globálního, tak i lokálního kontrastu. Toho je dosaženo rozšířením vstupu operátoru o hodnoty okolních pixelů. Výstup mapování je v tomto případě značně ovlivněn výběrem tvaru a velikosti okolí pixelu. Špatně zvolené okolí může způsobit nechtěné efekty, jako halo, nebo ztrátu barevných extrémů.

Frekvenční operátory

Frekvenční operátory jsou, stejně jako lokální operátory, zaměřené na zachování hran (detailů) a lokálního kontrastu. Jak napovídá název, úpravy u těchto metod probíhají ve frekvenční doméně. Zásadním poznatkem je, že hrany a lokální kontrast jsou zachovány pouze pokud pracujeme nezávisle s nízkými frekvencemi, které jsou spojeny s dopadajícím světlem, a s vysokofrekvenční složkou, která odpovídá detailům a odraženému světlu. Kvůli převodu

[9] Obrázek vytvořen na základě schématu, které uvádí Bočík [3].
[10] http://www.sim2.com/HDR/
[11] http://www.spheron.com

obrazu do frekvenční domény a zpět, patří tato skupina operátorů mezi nejnáročnější na výpočetní výkon.

Segmentační operátory

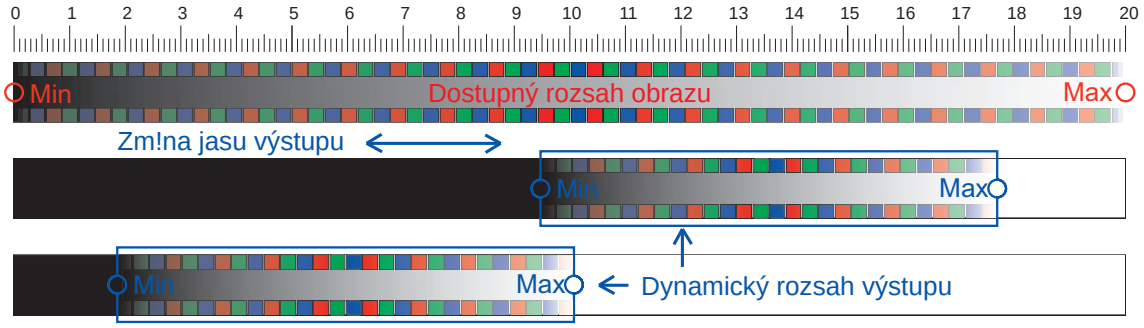
Nejnovějším přístupem k mapování tonality jsou segmentační operátory. Ostré hrany a prudké změny v kontrastu se často nacházejí podél okrajů velkých jednotvárných oblastí. Segmentační operátory proto nejprve rozdělí obraz na stejně velké oblasti, které jsou následně jednotlivě zpracovány globálním operátorem, a nakonec opět spojeny do výsledného obrazu. Výhodou tohoto přístupu je minimalizace změn v gamutu[12], zapříčiněná použitím jednodušších operátorů (většinou lineárních) na jednotlivé oblasti.

Podrobněji mapování tonality popisuje Freeman [5]. Popis tonálních operátorů s příklady použití pak uvádí Banterle a kolektiv [1]

2.3.3 Dynamická práce s obrazem

Mapování tonality umožňuje snížit dynamický rozsah, při zachování důležitých vlastností obrazu, což je vhodné pro úplný přenos HDR obrazu do LDR prostředí (časté použití ve fotografii). Náročnost na výpočetní výkon a složitost nastavení jsou zde ospravedlnitelné, protože jde o jednorázový převod.

Zaměřme se ale nyní na situaci, kdy je důvodem snížení dynamického rozsahu pouze prohlížení na LDR zařízení a výsledky nebudou dále využívány. Uživatel zároveň chce mít kontrolu nad převodem, aby mohl volit, které parametry obrazu budou zachovány. V takovém případě je třeba provést převod do LDR těsně před samotným zobrazením. To upřednostňuje použití kombinace výpočetně méně náročných algoritmů, které umožní například real-time převod videa (nutnost pro živé přenosy). Díky zapojení uživatele je možné vyhnout se snižování dynamického rozsahu celého obrazu. Místo toho můžeme zobrazit pouze část dostupného rozsahu, podobně jako při snímání scény s vysokým rozsahem jasů běžným digitálním snímačem. Další výhodou tohoto přístupu je, že se vyhneme problémům provázejícím mapování tonality, jako snížení lokálního kontrastu, nebo posterizace. Základní operací bude změna jasu, která zde bude fungovat jako posun výstupního obrazu v rámci dostupného dynamického rozsahu, jak je znázorněno na obrázku 2.5[13] [1].



Obrázek 2.5: Úprava jasu zobrazené části HDR obrazu.

Pro přiblížení síle tonálních operátorů by ale prohlížeč založený na tomto principu měl uživateli zpřístupnit i další operátory, například pro práci s gama křivkou obrazu, nebo

[12] Gamut v tomto případě reprezentuje množinu všech barev obsažených v obraze.
[13] Obrázek vytvořen na základě schématu, které uvádí Bočík [3].

saturací barev. Tyto další operátory ale budou použity přímo na zdrojový (HDR) obraz a z hlediska zobrazení na LDR zařízení je tedy nejdůležitější samotná práce s jasem.

Kapitola 3

Interakce člověka s počítačem

Interakce člověka s počítačem, nebo HCI z anglického *Human-computer interaction* se zabývá studiem, plánováním a návrhem způsobů komunikace mezi počítačem a člověkem, které vedou k co nejefektivnějšímu uspokojení uživatelských požadavků. Výpočetní technologie se v posledních letech staly téměř nepostradatelnou součástí každodenního života a návrháři pracující v oblasti HCI musí zvažovat několik faktorů: co lidé chtějí a očekávají, jaké jsou schopnosti a omezení lidského těla, jak zpracováváme smyslové vjemy, co je pro nás zábavné a příjemné [6]. Následující text není vyčerpávajícím výčtem všech aspektů HCI, ale spíše nástinem nejdůležitějších oblastí, kterými se HCI zabývá.

3.1 Komunikační kanály

Práci s počítačem lze popsat jako *obousměrnou výměnu informací mezi uživatelem a počítačem*. Aby tato výměna mohla smysluplně probíhat, je třeba aby komunikační kanály umožnovaly přenášet informace v reálném čase, reprodukovatelně a s co nejmenším zkreslením. Z hlediska využití těchto kanálů musí být dále zohledněna technická realizovatelnost, se kterou pak souvisí spolehlivost a cena příslušných periferních zařízení.

Přenos informací uložených v počítači k uživateli zprostředkují výstupní periferní zařízení, a uživatel je pak vnímá pomocí smyslů. Drtivá většina informací je v tomto směru předávána pomocí obrazu, což je umožněno díky jeho vysoké datové propustnosti, diverzitě možností reprezentace dat, a v neposlední řadě schopností přenášet několik nezávislých datových toků. Grafická data jsou často doplňována zvukem, který sice má nižší propustnost, díky přenosu pouze jednoho datového toku však může získat větší pozornost uživatele. Mobilní zařízení dále používají hmatovou odezvu ve formě vibrací, sloužící k upozorňování, nebo jako potvrzení uživatelských akcí (například při použití virtuální klávesnice). Zapojení dalších smyslů má velmi omezené využití (virtuální realita) a v současné době je problematicky technicky realizovatelné.

Opačný směr přenosu, od uživatele k počítači, je realizován pomocí vstupních periferních zařízení. Ta reagují na podněty od uživatele. Uživatelé zatím s výpočetní technikou interagují téměř výlučně pomocí pohybu. Typickou kombinací je nějaká forma klávesnice v kombinaci s polohovacím zařízením jako myš, touchpad, nebo trackball. Perspektivní možností, která by v budoucnu mohla v některých případech nahradit pohybové ovládání, je mluvená řeč. Rozpoznávání mluvené řeči se poměrně rychle vyvíjí a už dnes se objevují funkční aplikace, jako hlasové ovládání vyhledávače Google, nebo mobilní asistent Siri [16].

3.2 Pohybové modely

Jak bylo popsáno výše, pohyb těla a hlavně končetin je klíčový při interakci s výpočetní technikou. Při návrhu periferních zařízení a programového vybavení, které je využívá, je tedy nezbytné vzít v úvahu omezení, schopnosti a potenciál pohybového aparátu člověka. Za tímto účelem vznikly pohybové modely, které zjednodušují analýzu této, jinak velmi komplexní látky. Podle založení můžeme tyto modely rozdělit na Prediktivní, využívající ve velké míře matematický aparát, a Deskriptivní, které jsou spíše teoretické, a nabízí možnosti jak o dané problematice přemýšlet [4].

3.2.1 Prediktivní modely

Prediktivní, nebo také výkonnostní modely mají široké využití v různých odvětvích. Na poli interakce člověka s počítačem se používají k analytickému určení metrik lidské výkonnosti v dané situaci, bez nutnosti provádět praktické experimenty. Výsledné metriky umožňují prozkoumat část návrhu pouze v hypotetické rovině a získat přitom důležité informace, které by jinak byly přístupné pouze po implementaci celého systému. Jednotlivé modely se zaměřují na zjištění konkrétní metriky, jako doby potřebné k provedení dané úlohy (Keystroke-Level model), nebo rozhodovací doby (Hick-Hyman law).

3.2.2 Deskriptivní modely

Jednoduše řečeno, deskriptivní modely poskytují základ pro popis problému, nebo kontext pro jeho analýzu. Často se jedná jen o prostý výčet, nebo grafickou reprezentaci kategorií, nebo důležitých vlastností v rozhraní. Na rozdíl od prediktivních modelů, zde nezískáme konkrétní hodnoty, které by nám pomohly s rozhodováním. Místo toho nám deskriptivní modely pomohou prozkoumat danou problematiku a definovat důležitá rozhodnutí. Jako příklady můžeme uvést *třístavový model grafického vstupu* (3-state model of graphickal input), popisující vztah mezi polohovacími zařízeními a možnostmi interakce které poskytují , nebo *Key-action model*, který definuje kategorie tlačítek na klávesnici a jejich použití.

Více informací k této problematice, spolu s popisem vybraných modelů a příklady použití uvádí Carroll [4].

3.3 Uživatelská rozhraní

Uživatelské rozhraní, často označované jako UI z anglického *User Interface*, můžeme definovat jako prostředek, pomocí kterého může uživatel ovládat softwarovou aplikaci (Human-computer interface), nebo mechanické zařízení (Man-machine interface) [12]. Vzhledem k povaze práce se zaměřím pouze na první část této definice. Protože drtivá většina uživatelských rozhraní v počítači je grafického charakteru, dále v této práci používám termín uživatelské rozhraní, jako synonymum pro grafické uživatelské rozhraní (GUI).

3.3.1 Grafické prvky rozhraní

Primárním způsobem interakce u grafických rozhraní je nějaký typ polohovacího zařízení. Takové zařízení pracuje jako elektronický ekvivalent lidské ruky. Polohovací zařízení umožňuje práci s grafickými prvky rozhraní, často označovanými jako objekty. Objekty jsou často předdefinované ve vývojových prostředích pro GUI a jejich pomocí je definován vzhled rozhraní a možnosti interakce. Mezi hlavní zástupce patří následující [6].

Okna jsou oblasti umožňující umístění grafického obsahu nezávisle na zbytku systému. Uživateli je zpravidla umožněno měnit parametry těchto oblastí, jako rozměry, pozice v rámci obrazovky, nebo pořadí zobrazení.

Menu jsou hierarchické struktury běžně používané pro zpřístupnění příkazů dostupných v dané aplikaci. Příkazy jsou většinou rozděleny do skupin pro zjednodušení vyhledávání požadovaného příkazu. S menu se často setkáváme v podobě panelu v horní části obrazovky (Menu bar), nebo jako způsob nastavení vlastností dalších prvků (Context menu).

Ikony jsou piktogramy, nebo zmenšeniny obrázků, většinou reprezentující objekty souborového systému. Motiv ikony může odpovídat programu asociovanému s daným datovým typem, což uživatelům ulehčuje vyhledávání.

Ovládací prvky (Controls) jsou části rozhraní umožňující interkakci pomocí *přímé manipulace*. Každý prvek je vytvořen pro specifický způsob práce s rozhraním, ale většina je dostupná v téměř stejné podobě napříč vývojovými prostředími. To umožňuje udržovat rozhraní konzistentní i mezi různými typy zařízení. Příklady jsou tlačítka, posuvníky, nebo přepínače [6].

3.3.2 Možnosti hodnocení a testování

Ověřování vhodnosti návrhu a hledání chyb je vitální součástí vývojového procesu GUI. Návrhář, případně programátor často přemýšlí o funkcích rozhraní zásadně odlišným způsobem, než koncový uživatel. Uživatelské rozhraní se navíc musí přizpůsobit široké škále uživatelů. Následuje několik základních postupů, dostupných vývojovému týmu pro získání zpětné vazby, potřebné k řízení dalšího vývoje [14].

Expertní zhodnocení (Expert Reviews)

Výchozím bodem pro hodnocení bývá často předvedení produktu kolegům, nebo zákazníkům a vyhodnocování jejich reakcí. Přesto, že tento postup může poskytnout cenné náhledy, formálnější přístup, kdy je produkt předveden profesionálům v dané oblasti (nebo v oblasti uživatelských rozhraní), je většinou mnohem hodnotnější. Zřejmou nutností zde je mít takové odborníky ve vývojovém týmu, nebo přinejmenším jako konzultanty, což u menších projektů většinou není možné. Expertní zhodnocení mohou být použita téměř v jakékoli fázi projektu a poskytují rychlejší a přesnější odezvu, než uživatelské testování. Výsledkem hodnocení je pak formální popis nalezených nedostatků a doporučených změn, nebo diskuse s vývojovým týmem.

Testování použitelnosti (Usability testing)

Jedná se o studii rozhraní probíhající většinou v průběhu velké části vývoje a zahrnuje série uživatelských testů, sestavených pro testování vybraných vlastností rozhraní. Počet testovacích uživatelů se liší v závislosti na potřebách projektu. Menší skupiny uživatelů (3 - 6) umožňují bližší spolupráci s účastníky testu a použití technik jako *think aloud*, kdy uživatelé komentují svoji činnost a předávají tak velmi podrobný popis jejich zkušenosti s rozhraním. Rozsáhlejší testovací skupiny jsou naproti tomu vhodnější pro složitější rozhraní, protože poskytnou lepší pokrytí dostupných možností rozhraní a tím větší šanci nalézt případné chyby.

Průzkumy

Uživatelské průzkumy jsou založeny na analýze dotazníků zodpovězených rozsáhlou skupinou uživatelů, což jejich výsledkům dodává autoritu. Potřeba velkého množství respondentů však omezuje použití na průzkum trhu před začátkem návrhu, nebo na zlepšování kvality již nasazené aplikace. Nejčastější formou průzkumů jsou velmi krátké online dotazníky, které jsou levné a příliš neobtěžují respondenty.

Akceptační testování

U velkých projektů je zvykem sestavit před uzavřením smlouvy sadu testů a kritérií, sloužící k ověření požadovaných vlastností hotového produktu. V případě uživatelských rozhraní se často vyskytují vágně specifikované požadavky, jako „uživatelská přívětivost“. Takové požadavky je třeba přeformulovat za použití měřitelných veličin, jako čas nutný k zaučení s danou funkcionalitou, nebo procento výskytu chybných uživatelských vstupů.

Podrobnější popis zmíněných metod uvádí Shneiderman a Plaisant [14].

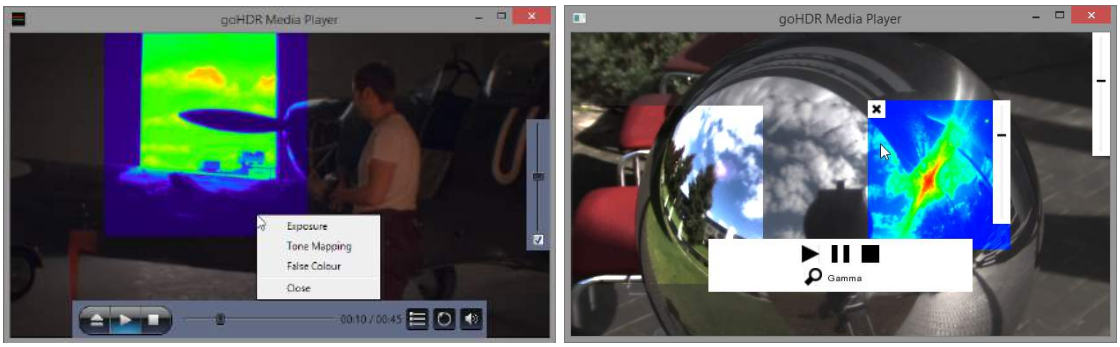
Kapitola 4

HDR přehrávače videa

Tato kapitola se soustředí na praktické nasazení metod zmíněných v kapitole 2 při přehrávání videa. Důraz je kladen na ovládací prvky a celkové chování přehrávačů. Dosažené závěry jsou použity dále ve fázi návrhu rozhraní.

4.1 Dosavadní tvorba GoHDR

GoHDR zatím vydala dvě verze přehrávače, založené na metodách popsaných v [1]. Vzhledem k velkým rozdílům mezi verzemi, způsobeným změnou vývojového týmu a značným časovým posunem, jsou zde zmíněny charakteristiky obou.



Obrázek 4.1: Přehrávače z produkce GoHDR.

První generace umožňuje úpravu jasu, mapování tonality, nebo zobrazení efektu false color (v tomto případě je každému pixelu přiřazena barva na základě intenzity světla). Úpravy jsou nastavitelné globálně, nebo ve vybrané oblasti. Velkou nevýhodou je zde sdílený posuvník pro úpravu jasu, který nutí uživatele smazat vybranou oblast kvůli změnám jasu v celém obraze.

Druhá (aktuálně používaná) verze opravila zmíněný nedostatek a zároveň zpřístupnila výběr několika nezávislých oblastí, se kterými může uživatel pohybovat. Úprava obrazu je pak rozšířena o práci s gama křivkou. Obě verze podporují pouze proprietární GoHDR formát HDR videa.

4.2 XDepth HDR video player

Jediným veřejně propagovaným konkurentem dosud zmíněných, je přehrávač XDepth firmy Trellis Management. Dostupnost tohoto produktu je však sporná. Beta testování první verze přehrávače je podle stránek produktu uzavřeno a nová verze je zatím nedostupná. Následující popis je založen pouze na informacích z demonstračních videí[1] a stránek produktu[2], protože se mi nepodařilo kontaktovat ani zástupce firmy.

Obrázek 4.2: XDepth HDR video player[1].

Kromě nastavení atributů obrazu jako kontrast, gama, nebo jas, srovnatelným s přehrávači goHDR, umožňuje přehrávač XDepth práci s barevnými kanály. Stránky produktu dále uvádějí kompatibilitu s HDR monitory, a manipulaci s videem zahrnující ořez, zoom a otáčení s podporou gest. Nevýhodou přehrávače je aplikace úprav na celý obraz najednou, což se, i při použití mapování tonality, negativně projeví například u scén s bodovými zdroji světla.

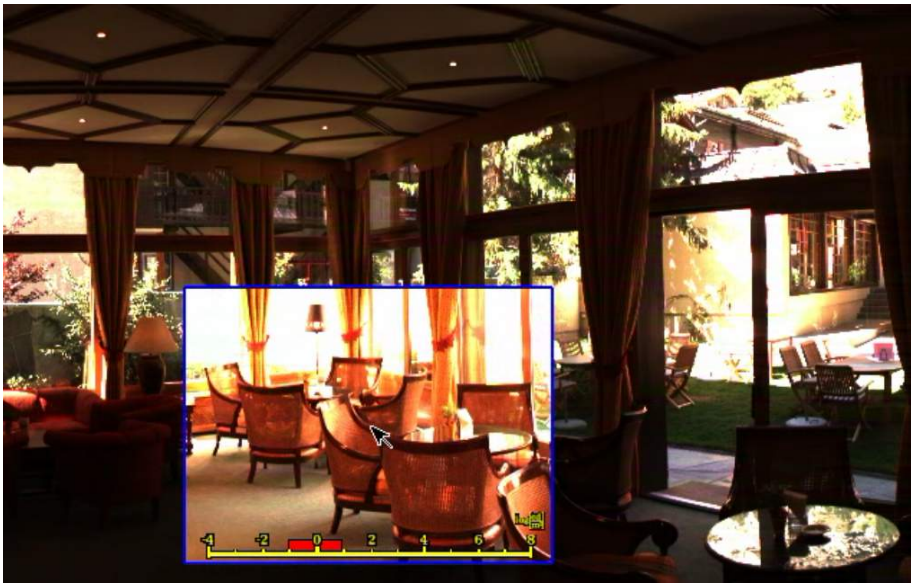
Existují verze pro Windows, Linux a Mac OS. Mezi podporované video formáty patří MPEG-1/2, DIVX a H.264 / MPEG-4 AVC a proprietární XDepth formát.

4.3 Přehrávač MPI Informatik

Tento přehrávač byl vytvořen pro výzkumné účely na Max-Planck-Institut für Informatik a přesto, že byl předveden už v roce 2004, dosud není k dispozici pro veřejnost. Zde zmíněné informace, včetně obrázku 4.3, pramení z článku [7], kde byl přehrávač předveden, a z demonstračního videa[3].

Tvůrci tohoto přehrávače se soustředili na přiblížení výsledného vzhledu obrazu původní scéně. K dispozici je proto několik tonálních operátorů s různými možnostmi nastavení. Pro úplnost disponuje přehrávač nástrojem pro výběr části dostupného dynamického rozsahu, odpovídající práci s jasem u ostatních zmíněných přehrávačů. Článek zmiňuje podporu OpenEXR a MPEG-4.

[1] https://www.youtube.com/watch?v=0keUqlZ_q6k
[2] http://www.xdepth.com/XDepthVideoHDR/index.html
[3] http://resources.mpi-inf.mpg.de/hdrvideo/supplemental/hdr_video.mov



Obrázek 4.3: Přehrávač MPI Informatik.

4.4 Současné možnosti využití

Ve fotografii se HDR poměrně rychle stalo fenoménem a v současné době jsou pořízení HDR snímku schopny nejen profesionální DSLR, ale i mobilní telefony. Situace s videem je složitější, protože záznam HDR videa je mnohem náročnější na snímací zařízení. Veřejná dostupnost HDR záznamů je téměř exkluzivně omezena na propagační záběry společností zabývajících se HDR. Zaměření zmíněných přehrávačů je tedy především výzkum, nebo úzká skupina společností, které mají možnost generovat vlastí HDR obsah.

Kapitola 5

Analýza požadavků a návrh

Tato kapitola se podrobně věnuje prvním fázím vývojového cyklu rozhraní. Pokusím se zde mimo jiné osvětlit důvody pro důležitější rozhodnutí, která bylo před začátkem samotné implementace třeba řešit, a jejich dopad na zbytek práce.

Jádrem této práce je uživatelské rozhraní přehrávače HDR videa vyvíjeného pod GoHDR, což je Britská společnost zabývající se nahráváním, zpracováním a reprodukcí HDR obrazu, pod vedením Profesora Alana Chalmerse. Vzhledem k nutnosti vytvoření verzí pro Linux, Mac Os a Windows se GoHDR přirozeně snaží udržet co největší část přehrávače multiplatformní. Na druhé straně je kladen velký důraz na rychlost a na to aby rozhraní přehrávače vizuálně „zapadlo“ do prostředí operačního systému, což upřednostňuje použití nativních vývojových prostředí pro GUI (frameworků). Na základě zmíněných argumentů vznikla multiplatformní knihovna LibScarlet, sloužící jako jádro aplikace, a k ní platformě závislá grafická rozhraní. Tato práce se zaměří právě na jedno z těchto rozhraní, konkrétně pro operační systémy Windows 7 a 8[1].

5.1 Základní požadavky

I bez formální specifikace můžeme z dosud zmíněných informací vyvodit následující seznam základních požadavků, který je směrodatný při návrhu rozhraní.

- Rychlost (responzivita)
- Profesionální vzhled
- Operační systém Windows
- Podpora OpenGL (LibScarlet)
- Podpora všech stávajících funkcí knihovny LibScarlet

V současné době nemá zpracování HDR videa hardwarovou podporu a LibScarlet je tím pádem sama o sobě náročná na výpočetní výkon. Rozhraní by proto mělo být co nejméně náročné, při zachování vzhledu a funkcionality hodné komerčního produktu. Nejvíce svazujícím požadavkem je pravděpodobně schopnost efektivně pracovat s OpenGL obsahem, protože nativní frameworky ve Windows jsou zpravidla založeny na DirectX. Poslední bod se pak vztahuje k rozhraní samotnému spíše než k použitým technologiím a je mu věnována pozornost dále v této kapitole.

[1] Podpora Windows XP by byla výhodou, největší důležitost mají ale novější verze systému.

5.2 Ovládací prvky

Kromě prvků, na které jsme zvyklí z běžných přehrávačů, jako jsou tlačítka „přehrát“, „pozastavit“, „stop“, nebo posuvník pro vyhledávání ve videu, bude do tohoto rozhraní třeba zakomponovat prvky specifické pro HDR. Jedná se o prvky sloužící k označení oblastí ve videu a výběr typu a intenzity filtrů, které budou na video použity (ukázka vzhledu a popis těchto prvků je zmíněn v kapitole 4). Uživatel musí mít možnost vymezit libovolný počet oblastí a vybírat typ a intenzitu filtru pro každou oblast zvlášť, nebo pracovat se skupinou vybraných oblastí. Zároveň bude možné vybrat globální filtr aplikovaný na celé video. Každou oblast bude možné libovolně přesouvat a měnit její velikost. GoHDR v blízké budoucnosti předpokládá rozšíření přehrávače o algoritmy pro sledování objektů ve videu a automatické „vylepšování“ obrazu ve vybraných oblastech. Tato rozšíření budou pravděpodobně vyžadovat značné úpravy v objektech reprezentujících vybranou oblast videa (mj. např. libovolný tvar ohraničení oblasti), což je třeba uvažovat v návrhu.

5.3 Grafický návrh

Při návrhu grafické stránky rozhraní byl kladen důraz na intuitivní a co možná nejjednodušší ovládání. Inspiroval jsem se proto ve velké míře u běžných přehrávačů videa, se kterými se drtivá většina uživatelů již setkala, a nebudou proto mít problém používat alespoň základní funkce.

5.3.1 Viditelnost ovládacích prvků

Při přehrávání HDR videa jsou ze zkušenosti nejčastěji používány prvky pro ovládání filtrů ve videu na rozdíl od videa s běžným dynamickým rozsahem (dále jen LDR, neboli Low Dynamic Range), kde se ovládací prvky používají obecně méně často a jde většinou o prvky na hlavním panelu (spuštění/pozastavení přehrávání, vyhledávání ve videu - seek). LDR přehrávače proto v režimu celé obrazovky zpravidla zobrazí, dosud skrytý hlavní panel, při každém pohybu kurzoru myši uvnitř okna přehrávače. Zde byl naproti tomu zvolen přístup selektivního zobrazování, kdy se prvky zviditelňují jednotlivě, například pokud se kurzor nachází v jejich blízkosti.

Obrázek 5.1: Návrh ovládacího prvku sloužícího pro výběr oblasti ve videu a ovládání filtru, který na ni bude aplikován.

5.3.2 Výběr oblasti ve videu

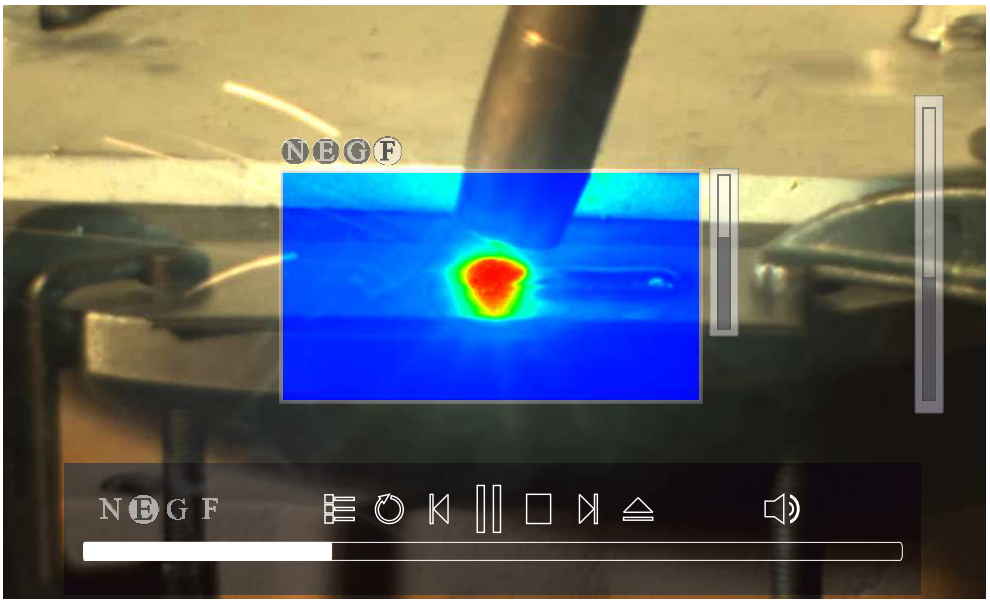
Na obrázku 5.1 je znázorněn první pokus o návrh ovládacího prvku reprezentujícího vybranou oblast ve videu. Oblast je ohraničena částečně průhledným rámečkem, který by měl být dostačující pro vymezení výběru, při relativně malém omezení viditelnosti videa pod ním. Sada přepínačů v horní části prvku slouží k výběru filtru, který bude na oblast aplikován, a posuvník napravo umožní volit intenzitu efektu filtru.

5.3.3 Hlavní ovládací panel

Na obrázku 5.2 vidíme dvě varianty této části rozhraní, jejichž vhodnost značně závisí na výsledném způsobu použití přehrávače a na uživatelských preferencích. Rozhodnutí mezi nimi proto bylo odloženo do doby, kdy bude přehrávač disponovat většinou specifikovaných funkcí a bude ho možné prezentovat a hodnotit mimo testovací prostředí. Hlavnímu panelu

Obrázek 5.2: Porovnání volně pohyblivého (vlevo) a ukotveného ovládacího panelu u přehrávače VLC.

ukotvenému ve spodní části obrazovky konkuruje oddělený ovládací prvek. Nezávislý panel by bylo možné s výhodou používat na sestavách s více monitory, kde umožňuje pohodlné ovládání přehrávače, přičemž nepřekáží v obraze ani v režimu celé obrazovky. Někteří uživatelé však stále preferují první možnost, protože panel je vždy „na svém místě“ a nemusí se o něj starat.



Obrázek 5.3: Návrh a rozmístění ovládacích prvků.

5.3.4 Celkový vzhled

Obrázek 5.3, demonstruje všechny zmíněné ovládací prvky a jejich předběžné rozmístění, které vychází ze starších přehrávačů GoHDR. Předpokládá se jeho další vývoj na základě uživatelských testů v průběhu implementace.

5.4 Volba frameworku

Prvním naivním pokusem o zobrazení LibScarlet ve Windows byl port testovací verze z Linuxu. Jednalo se o kombinaci SDL[2] a LibRocket[3], která elegantně řešila zobrazení OpenGL. Zmíněné řešení jsme však museli zavrhnout kvůli celé řadě problémů, jako velmi pomalé reakce ovládacích prvků, fragmenty v obraze a pomalá adaptace rozhraní při změně velikosti okna, nebo problematická změna vzhledu okrajů okna a horní lišty. Tento výsledek nás utvrdil v přesvědčení, že bude výhodnější použít skutečně nativní framework (postavený na DirectX), který nebude podobnými neduhy trpět, a to i za cenu že bude třeba věnovat zvláštní pozornost zobrazení OpenGL obsahu. Aktivní znalost napříč vývojovým týmem a dostupnost ve všech cílených verzích Windows následně rozhodly o použití podmnožiny *.NET frameworku*, WPF (Windows presentation foundation[4]).

5.5 Windows presentation foundation

Windows presentation foundation, dříve označovaný jako Avalon, je systém pro tvorbu a zobrazování grafických rozhraní v prostředí Windows. Jedná se o nástupce systému WinForms se zaměřením na graficky bohatá rozhraní.

Kód v WPF se obvykle skládá ze dvou částí. Grafické prvky jsou definovány pomocí značkovacího jazyka XAML a logiku aplikace (tzv. Code behind) popisuje interpretovaný jazyk jako Microsoft Visual Basic, nebo C#.

Knihovny potřebné pro běh většiny WPF aplikací jsou součástí operačních systémů Windows Vista a Windows Server 2008, nebo novějších. Dále je možné je doplnit do Windows XP SP2/SP3 a Windows Server 2003.

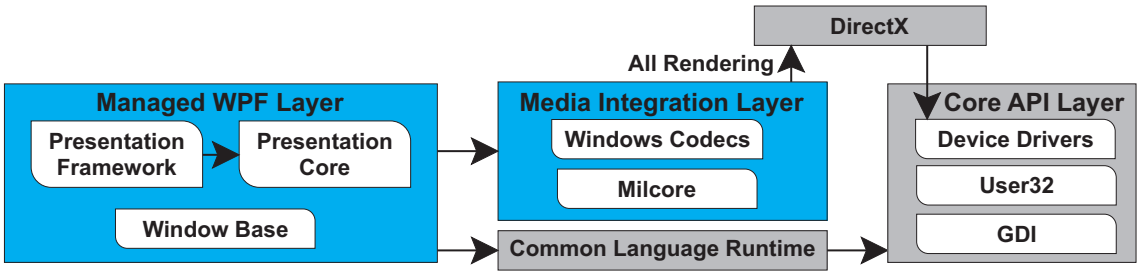
5.5.1 Přehled architektury

WPF je sada modulů, rozdělených do dvou vrstev (*Managed WPF layer* a *Media Integration layer*), jak je vidět na obrázku 5.4[5]. WPF dále využívá funkce jádra systému, ve schématu označované jako *Core API Layer*) [15].

Managed WPF layer je sada tříd a typů postavených na .NET frameworku, což znamená, že je interpretovaná pomocí CLR (Common Language Runtime). Tato vrstva obsahuje tři složky, které dohromady tvoří WPF API (Aplication Programming Interface neboli rozhraní pro tvorbu aplikací).

- **Window Base Assembly** obsahuje základní typy potřebné k tvorbě WPF programů, včetně tříd Aplication, nebo Window. Zajišťuje také základ pro práci s vlákny a se závislostí atributů mezi objekty.

[2] Simple DirectMedia Layer http://www.libsdl.org/
[3] http://librocket.com/
[4] http://msdn.microsoft.com/en-us/library/ms754130(v=vs.110).aspx
[5] Obrázek byl vytvořen na základě [15]



Obrázek 5.4: Schéma architektury WPF.

- **Presentation Framework** poskytuje funkcionalitu potřebnou k sestavení aplikace, jako data binding, práce se styly, společný základ pro ovládací prvky a další.
- **Presentation core** slouží jako přechod mezi interpretovaným kódem a MILCore, pro který poskytuje veřejné rozhraní. Zároveň je základem pro zobrazování WPF a zpřístupňuje třídy pro tvorbu takzvaného *visual tree*. *Visual tree* je strom obsahující grafické prvky a instrukce pro jejich vykreslování.

Media Integration layer, nazývaná také **MILCore** kvůli její hlavní součásti, přijímá instrukce pro zobrazování uživatelského rozhraní a překládá je na data srozumitelná pro DirectX. DirectX poté zajišťuje práci s grafickou kartou. Tato vrstva není z důvodu optimalizace interpretovaná.

Core API Layer zahrnuje komponenty jádra systému jako Kernel, User32 (práce s pamětí a oddělení procesů), ovladače periferních zařízení, a další [15].

5.5.2 Důležité koncepty

Data binding (datová vazba) je proces zajišťující spojení mezi uživatelským rozhraním a logikou aplikace. V případě správného nastavení jsou všechny navázané objekty informovány o případných změnách hodnoty sledované atributy. Pro správnou funkci je nutné, aby sledovaný objekt implementoval rozhraní *INotifyPropertyChanged*. Pomocí datových převodníků (*IValueConverter*), je možné vázat normálně nekompatibilní datové typy, nebo odvozovat hodnotu atributy z několika zdrojů [15].

Commands (příkazy) jsou vstupním mechanizmem umožňujícím polymorfismus. Jejich hlavním cílem je oddělení sémantiky požadavku od logiky, která ho bude vykonávat a od objektů, které ho vyvolávají. Příkazy dále zajišťují mechanizmus pro zjištění, zda je daná akce v daný okamžik proveditelná. Samotný příkaz obsahuje pouze identifikátor a případný popis. Objekt, který by mněl na příkaz reagovat, použije *CommandBinding* k označení metody obsahující logiku pro zpracování příkazu a metody pro kontrolu, zda je možné příkaz provézt. V případě vyvolání příkazu vznikne událost (*RoutedEvent*), která prochází stromem prvků aplikace dokud nenarazí na *CommandBinding* [15].

Styly a šablony

Grafické prvky ve WPF jsou rozděleny na logiku (Code Behind), definující chování, a šablony (template), které obsahují jejich grafický strom. Šablony přímo určují součásti, ze chte-

rých se jednotlivé prvky rozhraní skládají, jejich rozmístění a relativní velikost. Umožňují také nastavit grafické reakce na uživatelské podněty, nebo způsob grafické reprezentace dat objektu. Celkový vzhled jednotlivých prvků je možné měnit i za chodu aplikace úpravou, nebo úplnou výměnou šablony. Základní vzhled určují výchozí šablony (Default template), jejichž výběr závisí na používaném vzhledu (theme) operačního systému. Styly pak přidávají možnost nastavit vybrané atributy celé skupině příbuzných objektů [15].

5.5.3 XAML

Extensible Application Markup Language, zkráceně XAML, je značkovací jazyk založený na XML, vytvořený pro sestavování a inicializaci hierarchických objektů platformy *.NET*. V současnosti se XAML používá kromě WPF také ve Workflow Foundation a Silverlight.

Všechny třídy ve WPF mají konstruktory bez parametrů a soustřeďují se na použití atribut, aby byly snadno použitelné s XAML. Jakýkoli zápis v XAML je možné nahradit kódem v C#, nebo Visual Basic. V praxi se ale kvůli větší přehlednosti používá kombinace XAML (většinou definuje hlavně grafický strom) a C#.

Pro zjednodušení práce s XAML slouží *XAML Designer*, který je součástí Visual studia a samostatná aplikace *Expression Blend* zaměřená čistě na grafický návrh [10].

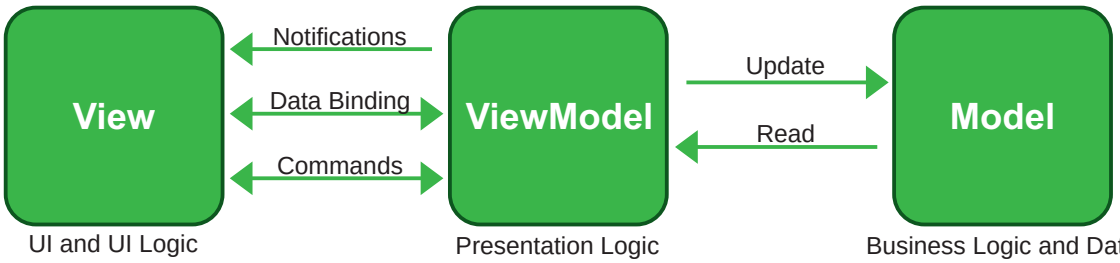
5.5.4 Model-View-ViewModel

Model-View-ViewModel, dále jen MVVM, je varianta návrhového vzoru *Presentation Model*. Je optimalizovaná pro využití základních konceptů WPF, popsaných v předchozí části této práce [9].

View má na starost strukturu a vzhled toho co uživatel uvidí na obrazovce. *Code-behind* ideálně obsahuje pouze konstruktor, případně chování, které se špatně vyjadřuje ve formě XAML, jako složité animace, nebo interakce s dalšími grafickými prvky.

View Model obsahuje prezentační logiku a stavové informace. Nemá možnost přímo přistupovat k *View*, ani nemá informace o jeho typu. *View Model* zpřístupňuje atributy a příkazy, které může *View* využívat a upozorňuje na případné změny. Vystavené atributy a příkazy definují funkcionalitu rozhraní, ale *View* určuje, jakým způsobem bude zpřístupněna uživateli.

Model zahrnuje vnitřní fungování aplikace a statická data. Pravidlem je, že model nesmí mít informace o stavu ovládacích prvků. Veškerá komunikace s uživatelem probíhá přes *View Model* a rolí modelu je ověření vstupů a zajištění konzistence dat.



Obrázek 5.5: Schéma návrhového vzoru MVVM.

Jak je vidět na obrázku 5.5[6], *View* komunikuje s *View Model* přes příkazy a upozornění na změny hodnot. *View Model* předává potřebná data z modelu uživatelskému rozhraní

[6] Schéma bylo vytvořeno na základě [9]

a změny způsobené uživatelským vstupem propaguje zpět.

Tento vzor není vhodný pro malé projekty, kde se výrazně projeví zvýšení režie, a pro velmi rozsáhlá rozhraní, u kterých by *data binding* zabíral velké množství operační paměti. Podrobněji návrhový vzor popisuje [9].

Kapitola 6

Implementace a testování

Tato kapitola se zabývá důležitými částmi procesu realizace návrhu. Je zde popsáno řešení větších překážek v průběhu implementace a postup testování spolu s jeho vlivem na návrh. V závěru kapitoly je prezentován výsledný vzhled rozhraní a možnosti dalšího rozšiřování celého projektu.

6.1 Implementace

V návaznosti na zvolený framework proběhla implementace ve vývojovém prostředí Microsoft Visual studio 2013 Premium s aplikací Expression Blend pro úpravu stylů a tvorbu vzhledu vlastních (Custom) UI prvků. Popisky tlačítek pak vznikly v programu Corel Draw X6.

6.1.1 Zobrazení OpenGL obsahu v prostředí WPF

Povaha zdrojů informací k dané problematice mě přiměla experimentálně otestovat všechny nalezené možnosti, pro získání aktuálních informací o každém přístupu. Podařilo se mi replikovat dvě následující metody.

První spočívá v renderování OpenGL do bufferu který nebude přímo zobrazen, ale místo toho se jeho obsah přenese do prostředí WPF jako bitová mapa a zobrazí se jako statický obrázek. Tento postup zapouzdřuje knihovna *SharpGL*[1].

Implementačně složitějším, ale potenciálně rychlejším řešením je použít WindowsFormsHost, nebo HwndHost v kombinaci s toolkitem *OpenTK*[2]. Tyto prvky umožní zobrazení OpenGL v části WPF okna. WindowsFormsHost/HwndHost vytvoří „podokno“ (hosted child window), do kterého můžeme pomocí OpenTK vykreslovat OpenGL obsah. Použití tohoto způsobu integrace bohužel přináší značná omezení, jako nemožnost zobrazení dalších WPF prvků nad samotným objektem WindowsFormsHost/HwndHost v rámci jednoho okna[3], nebo změny stylu okna[4]. Možnosti řešení zmíněných omezení jsou podrobně rozepsány dále v této kapitole.

[1] http://sharpgl.codeplex.com/

[2] http://www.opentk.com/

[3] Téma podrobně rozebírá Need [11].

[4] Atributy *WindowStyle* a *AllowsTransparency*, umožňující práci se vzhledem okrajů a horní lišty okna, není při použití WindowsFormsHost/HwndHost možné měnit.

6.1.2 Interakce C++ a WPF

Další krokem bylo zajištění spolupráce mezi neinterpretovaným (unmanaged) C++ a WPF, pracujícím pouze pomocí CLR (Common Language Runtime). Drtivá většina interakcí mezi LibScarlet a uživatelským rozhraním je zahájena ze strany rozhraní, protože jde o reakce na uživatelské povely (výběr souboru k přehrávání, zvýšení hlasitosti, ...). Komunikaci v opačném směru (chyby, konec souboru, ...) je pak možné nahradit voláním metod pro kontrolu stavu LibScarlet (detekce chyb) a zvýšením úrovně samostatnosti rozhraní (detekce konce souboru na základě udané délky). Tento problém tedy můžeme zjednodušit na jednostranné volání metod napsaných v C++ z prostředí WPF.

K tomu účelu je určen balíček *InteropServices*. Funkce *DllImport* umožňuje načíst funkce, nebo celé třídy z dynamicky linkovaných knihoven a používat je v interpretovaném prostředí. Pro ověření kompatibility knihovny SharpGL a takto importovaných funkcí jsem vytvořil testovací program obsahující jednoduchou třídu v C++ a WPF okno s demo animací v OpenGL. Program najdete na přiloženém DVD (`Code\WPF_importdll_GL`). Voláním metody *Draw()*, importované pomocí *DllImport*, je do animace přidána bílá úsečka.

6.1.3 Další pokusy s OpenTK

Výkonnostní problémy při použití SharpGL na slabších počítačích mě vedly k prozkoumání možností použití druhé zmíněné varianty integrace OpenGL do WPF. Jak vysvětluje Need [11], obsah hostovaný pomocí WindowsFormsHost, nebo HwndHost, se vždy vykresluje přes ostatní prvky grafického stromu WPF. Z důvodu pochybností vyslovených na online diskusních fórech Microsoftu jsem experimentálně ověřil, že je překryta i tzv. *Adorner Layer*, což je nejvyšší programově dostupná vrstva WPF. Další úsilí v tomto směru jsem proto směřoval k synchronizaci pohybu a rozměru dvou oken, kde spodní z nich obsahuje WindowsFormsHost a z většiny průhledné horní okno potom vykresluje uživatelské rozhraní.

Synchronizaci oken značně zjednodušil koncept hierarchie oken ve WPF. Nastavení vztahu mezi okny pomocí atributu *Owner*, kdy okno A (Owner) je nadřízené oknu B (Owned), zajistí následující chování[5].

- Při minimalizaci A je minimalizováno i B.
- Minimalizace B nemá vliv na A.
- Při maximalizaci A je B obnoveno do popředí.
- A nemůže nikdy překrývat B.
- Zavření A způsobí i zavření B.
- Pokud nebylo okno B otevřeno pomocí *ShowDialog*, nestane se modálním.

Ze 4. bodu vyplývá, že okno s videem bude nadřízené oknu s rozhraním. Pokud dále vyřadíme okno s rozhraním z výběru v nabídkách přepínání oken (alt+tab, win+tab) a hlavního panelu, WPF ošetří většinu situací.

Použitý koncept by upřednostňoval použití spodního okna typu Win32, díky čemuž by odpadla potřeba WindowsFormsHostu. Synchronizace takového okna s oknem rozhraní je

[5] http://msdn.microsoft.com/en-us/library/system.windows.window.owner(v=vs.110).aspx

však mnohem problematičtější (část synchronizačního mechanizmu v C++) a nechává více prostoru pro chyby.

I přes zjednodušení situace pomocí správy oken z WPF a testování výsledné synchronizace oken není zajištěna správná funkčnost ve všech situacích. Aktuální nastavení se totiž opírá o jisté předpoklady o interakci prostředí systému s okny. Problémy mohou způsobit například budoucí aktualizace systému, nastavení správy oken, které nebyly zváženy, nebo software třetích stran. Výsledek můžete najít na přiloženém DVD (`code\GoHDR -- Mockup1_openTK`).

6.1.4 Grafická stránka

Jako základ barevné kompozice byla použita sada stylů *ExpressionDark*[6], dostupná jako součást toolkitu *Silverlight*[7]. V rámci postupného vývoje rozhraní byla ale velká většina těchto stylů upravena, nebo úplně nahrazena a její hlavní přínos tak spočíval hlavně v definici jednotné sady barevných štětců (*TextBrush*, *ControlBackgroundBrush*, atp.) použitelných napříč rozhraním.

Ve velké míře jsem používal třídy odvozené z *UserControl* což umožnilo zapouzdřování částí rozhraní s provázanou logikou. Instance samotné třídy *UserControl* pak pomohly v situacích, kdy bylo třeba animovat celou část rozhraní na základě chování uživatele, jako tomu bylo například u hlavního ovládacího panelu. V režimu celé obrazovky se tento panel automaticky skrývá a neviditelná *UserControl* zajišťuje jeho opětovné vysunutí při přiblížení kurzoru myši.

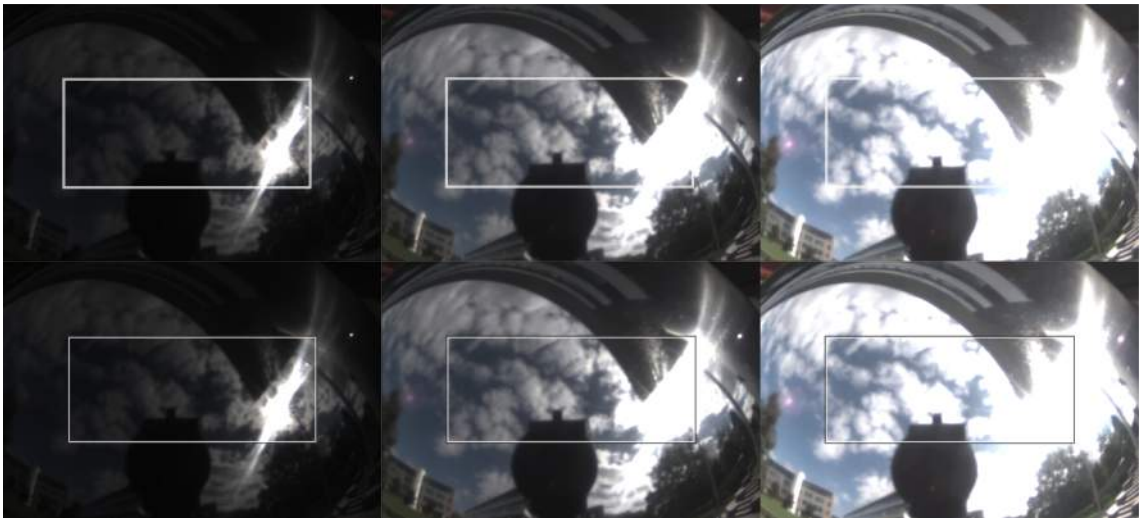
Systém šablon a stylů WPF umožnil úpravu vzhledu a do jisté míry i chování většiny potřebných ovládacích prvků. Jednou z výjimek byl však posuvník (Slider). Složitost základní šablony a stylu (přes. 250 řádků) značně zkomplikovala i nastavení vzhledu. Hlavním problémem bylo ale chování, kdy kliknutí do dráhy posuvníku pouze posune ukazatel o nastavený krok směrem ke kurzoru, místo jeho přemístění na místo kliknutí, jak jsou uživatelé zvyklí. Pro vyhledávání ve videu, nastavení hodnot filtrů a úpravu hlasitosti bylo proto třeba vytvořit nový ovládací prvek.

Vývoj návrhu v průběhu implementace

Průběžné uživatelské testy si pochopitelně vynutily úpravy navrženého vzhledu. Většina těchto úprav vyplývala z hledání kompromisu mezi rozumnou viditelností ovládacích prvků a jejich „rušivostí“ při sledování videa. Tohoto kompromisu bylo dosaženo pomocí dynamické práce s průhledností. Prvky se objevují v přítomnosti kurzoru myši, nebo pokud je jejich hodnota změněna klávesovou zkratkou a jinak jsou většinou skryty. Využití barevného kontrastu, jak je vidět na obrázku 6.1, navíc umožnilo udržovat prvky částečně průhledné (50% – 60% krytí), při zachování jejich funkce.

Další úpravy si vyžádala změna specifikace zvýšením počtu video filtrů. Způsob výběru těchto filtrů se změnil z přepínacích tlačítek (radio button), na výběrové pole (combo box). Tato změna zároveň výrazně zjednoduší další úpravy seznamu filtrů, které nyní budou spočívat pouze v úpravě textového seznamu a výčtového typu, namísto vytváření nového tlačítka. Další možností by bylo použití kontextového menu. Ale podle zkušeností GoHDR s jejich prvním přehrávačem, většina uživatelů takovou volbu samostatně nenašla. Práce s přehrávačem byla navíc, při použití více různých filtrů, značně nepřehledná.

[6] http://wpfthemes.codeplex.com/
[7] http://www.microsoft.com/silverlight/



Obrázek 6.1: Porovnání prvního návrhu ohraničení výběru videa (nahoře) a finální verze využívající kontrast (dole), v různých světelných situacích.

Při manipulaci s oknem aplikace měli uživatelé často problém s odděleným ovládacím panelem, který „překážel“, nebo naopak nebyl k nalezení. Na základě těchto reakcí byl implementován „lepivý“ (sticky) efekt, který v případě potřeby umožní připojit panel k oknu s videem. Problémy s hledáním ztraceného panelu dále řeší možnost ponechat panel stále viditelný i v režimu celé obrazovky, a klávesová zkratka, která panel přesune do výchozí pozice ve spodní části okna s videem.

6.1.5 Logická struktura

Použití návrhového vzoru Model-View-ViewModel mi umožnilo plně se soustředit na grafickou stránku rozhraní bez nutnosti řešení datových závislostí. Zapouzdření jádra aplikace (LibScarlet) zjednodušuje použití zvoleného vzoru, který takto nevnese do aplikace tolik nadbytečného kódu a s ním další režie.

Jako View Model slouží třídy *PlayerInterface* a *CommandLibrary*. *PlayerInterface* poskytuje rozhraní k funkcím *LibScarlet*, a uchovává stavové informace ovlivňující chování rozhraní, jako délku aktuálně přehrávaného videa, nebo zda probíhá přehrávání. Jak napovídá název, *CommandLibrary* obsahuje definice drtivé většiny příkazů použitých napříč rozhraním.

6.2 Výsledný vzhled a chování

Rozhraní je laděno do tmavých odstínů, protože méně unavují oči a ve videu většinou působí méně rušivě než světlé barvy. Ve velké míře jsou pak používány kontrasty, které umožňují práci s průhledností při zachování zřetelnosti ovládacích prvků. Interaktivita rozhraní je zajištěna reakcemi na přítomnost myši, nebo změny hodnot pomocí klávesových zkratek.

Integrace s LibScarlet proběhla u dvou výsledných verzí rozhraní. Jak je vidět na obrázcích 6.2 a 6.3, rozdílem obou verzí je pohyblivost hlavního panelu. První verze se řídí příkladem většiny běžných přehrávačů a hlavní panel je zde ukotven ve spodní části aplikace. U druhé verze je možné s panelem nezávisle manipulovat, nebo ho v případě potřeby

Obrázek 6.2: Výsledný vzhled rozhraní s ukotveným ovládacím panelem.

„připnout“ k oknu s videem, čemuž napomáhá lepivý efekt.

V době odevzdání této práce byla knihovna LibScarlet stále rozpracovaná a nebylo proto možné přiložit funkční verzi celého přehrávače. Vývoj tohoto rozhraní byl však podpořen stálou zpětnou vazbou od vývojového týmu GoHDR, která by měla zajistit kompatibilitu a připravenost na dosud neimplementované funkce jádra aplikace.

Uživatelská přívětivost je dále podpořena více způsoby pro přístup k většině funkcí. V některých případech jde pouze o doplnění tlačítek klávesovými zkratkami, například otevření souboru s videem je ale možné provést dvěma různými tlačítky (v hlavním panelu a v editoru playlistu), pomocí *drag-and-drop*, nebo klávesovou zkratkou ctrl + O.

6.3 Testování

Za účelem prezentace a testování některých ovládacích prvků bylo třeba, kvůli velmi omezené funkčnosti LibScarlet, některé funkce pro práci s videem simulovat. Pro testování oblastí ve videu bylo například použití různých video efektů napodobeno pomocí několika variant stejné fotografie po aplikaci různých filtrů. Vybraná oblast ve videu pak zobrazuje některou z variant fotografie v závislosti na vybraném efektu[8]. Při vývoji a testování prvků společných s běžnými přehrávači videa, byl pak použit prvek MediaElement, což je součást WPF fungující jako jednoduchý přehrávač.

[8]Tento testovací režim je přístupný po sestavení aplikace s kompilačním symbolem STATIC_IMAGE

Obrázek 6.3: Výsledný vzhled rozhraní s oddělitelným ovládacím panelem.

6.3.1 Programové testy

Tato část testování se zaměřila na programovou funkčnost aplikace a její náročnost na výpočetní zdroje. Použití návrhového vzoru MVVM umožnilo oddělení testů logiky rozhraní od grafických prvků. To se ukázalo být zvláště výhodné při ověřování korektního připojení rozhraní k LibScarlet.

Hardwarová náročnost byla měřena pomocí programu Resource Monitor, který je součástí systému Windows, a profilovacích funkcí Visual Studia. Tyto testy probíhaly na procesorech Intel Core i3 a i5 s taktem 2,2GHz, při obnovovací frekvenci OpenGL obsahu 30 FPS a aktivním používání prvků rozhraní. Tabulka 6.1 demonstruje náročnost částí aplikace a porovnání se staršími přehrávači GoHDR.

	CPU	**RAM**
Samotné rozhraní	4%	61MB
Rozhraní s SharpGL	19%	67MB
Rozhraní s OpenTK	12%	76MB
Kompletní aplikace	29%	198MB
Přehrávač GoHDR 1	26%	384MB
Přehrávač GoHDR 2	32%	231MB

Tabulka 6.1: Náročnost částí aplikace na výpočetní zdroje.

6.3.2 Uživatelské testy

Absence většiny funkcí jádra aplikace značně omezila možnosti uživatelského testování. Uživatelé byly konfrontováni pouze s testovacími verzemi simulujícími určité funkce, které byly popsány výše. Tento postup byl vhodný pro přizpůsobení jednotlivých ovládacích prvků, ne však pro optimalizaci jejich interakcí v rámci rozhraní jako celku. Pro tento účel pak byly navrženy dvě následující skupiny objektivních testů, které bude možné provést po dokončení základních funkcí LibScarlet.

Práce se soubory a vyhledávání ve videu

Uživatel dostane několik videí označených pouze čísly a bude mít za úkol doplnit tato čísla do korespondujících kolonek v tabulce obsahující krátký popis videa. Dále může být požadováno u každého videa určit délku, případně čas ve kterém ve videu proběhne specifikovaná událost. U tohoto typu testů je pro zjednodušení možné používat LDR videa.

Použití HDR filtrů

V rámci těchto testů bude uživatel hledat jev rozpoznatelný pouze při použití některých filtrů[9]. Příkladem může být hledání času a části obrazovky, ve kterých se ve videu objevil nejjasnější bod. Například u videa obsahujícího náhodné exploze[10], je pro splnění takového úkolu třeba použít efektu *false-color*.

6.4 Další vývoj

V budoucnu se předpokládá adaptace tohoto rozhraní na nové funkce LibScarlet, mezi které by mělo patřit například již zmíněné sledování objektů ve videu spojené s mapováním tonality v jejich okolí za účelem zlepšení viditelnosti. Můžeme také očekávat rozšíření přehrávače pro práci s HDR displeji, které začala GoHDR nedávno používat[11]. Předběžně se předpokládá souběžné používání běžných a HDR displejů, kde by HDR displeje zobrazovaly klon okna přehrávače, bez ovládacích prvků v obraze, s možností přiblížení části videa.

S drobnými úpravami by bylo možné přehrávač používat i jako jednoduchý prohlížeč HDR fotografií. Pro toto nasazení by však bylo vhodné doplnit možnost alespoň základní manipulace s prohlíženými fotografiemi, jako rotace a přiblížení.

Kromě zmíněných funkčních rozšíření má smysl uvažovat o prvcích pro zlepšení *user exprience*, jako sada barevných stylů které by změnily nádech celého rozhraní, nebo volitelný vzhled tlačítek.

[9] Pro zmíněné testy jsou vhodná videa GoHDR http://gohdr.com/website/samples-hdr-footage/

[10] https://files.warwick.ac.uk/alanchalmers/files/goHDRv_dynamicIBL.hdrv

[11] GoHRD v současné době vlastní několik prototypů HDR displejů *Solar 47 firmy SIM 2* (http://www.sim2.com/HDR/)

Kapitola 7

Závěr

Tato práce se zabývá zpracováním HDR obrazu, se zaměřením na návaznost HDR na současné technologie. Praktickým cílem pak byl vývoj uživatelského rozhraní pro ovládání přehrávače HDR videa. Tento cíl byl splněn a výsledné rozhraní je k dispozici na přiloženém DVD.

Pro zvládnutí zadaného úkolu bylo nejprve třeba identifikovat současné možnosti zobrazení HDR, kterým se spolu s jejich začleněním do multimediálního průmyslu věnuje začátek práce. Analýza těchto informací pak vedla ke specifikaci funkcionality a návrhu ovládacích prvků popsaných v kapitole 5. Výsledné rozhraní bylo implementováno v prostředí WPF v několika verzích, lišících se vzhledem a způsobem zobrazení *OpenGL* obsahu (*SharpGL*, nebo *OpenTK*). Ve verzích založených na *SharpGL* byla zajištěna interakce s funkcionalitou poslední dostupné verze knihovny LibScarlet. Jednotlivé ovládací prvky byly otestovány pomocí prototypů rozhraní, simulujících navržené funkce LibScarlet. Návrh rozsáhlejších uživatelských testů je, spolu se zmíněnými testovacími prototypy, popsán v kapitole 6.

Za hlavní úspěchy považuji rychlost ovládání výsledného rozhraní, zřetelnou zvláště při práci s oblastmi ve videu, a vysokou úroveň interaktivity ovládacích prvků, která rozhraní odlišuje od jeho předchůdců. Naopak, jednou z největších výzev se ukázalo být nalezení vhodných způsobů simulace funkcí knihovny LibScarlet za účelem testování prvků rozhraní.

Celý projekt má velký potenciál pro budoucí rozšíření. Čistě z pohledu uživatelského rozhraní by se jednalo například o práci se vzhledem (sada barevných stylů, nastavitelný vzhled tlačítek), nebo o jazykové lokalizace. Další rozšíření by pak vyžadovaly i úpravy v samotné LibScarlet. Mezi takové patří libovolný tvar vybraných oblastí pro efektivní sledování objektů, nebo další okno přehrávače s novou sadou ovládacích prvků pro zobrazení na HDR monitorech.

Literatura

[1] Banterle, F.; Artusi, A.; Debattista, K.; aj.: *Advanced High Dynamic Range Imaging: Theory and Practice*. AK Peters (CRC Press), 2009, ISBN 978-156881-719-4.

[2] Bloch, C.: *HDRI pro fotografy a počítačové grafiky: High Dynamics Range Imaging*. Encyklopedie - grafika a fotografie, Zoner Press, 2008, ISBN 9788074130014.
URL http://books.google.cz/books?id=10y5tgAACAAJ

[3] Bočík, A.: *Velká kniha HDR fotografie, kouzlo fotografií s vysokým dynamickým rozsahem*. Computer Press, druhé vydání, 2011, ISBN 978-80-251-3367-5.

[4] Carroll, J.: *Hci Models, Theories, and Frameworks: Toward a Multidisciplinary Science*. The Morgan Kaufmann Series in Interactive Technologies Series, Morgan Kaufmann Publ., 2003, ISBN 978-155860-808-5.
URL http://books.google.cz/books?id=gR3Imgvr5dYC

[5] Freeman, M.: *Základy HDR – Fotografie a vysoký dynamický rozsah*. Zoner Press, 2008, ISBN 978-80-86815-95-4.

[6] Galitz, W. O.: *The Essential Guide to User Interface Design*. John Wiley and Sons, Inc., druhé vydání, 2002, ISBN 0-471-084646.

[7] Mantiuk, R.; Krawczyk, G.; Myszkowski, K.; aj.: Perception-motivated High Dynamic Range Video Encoding. *ACM TRANSACTIONS ON GRAPHICS*, ročník 23, 2004: s. 733–741.

[8] Mantiuk, R.; Mantiuk, R.; Tomaszewska, A.; aj.: Color Correction for Tone Mapping. *Computer Graphics Forum (Proc. of EUROGRAPHICS 2009)*, ročník 28, č. 2, 2009: s. 193–202.
URL https://www.cs.ubc.ca/~heidrich/Papers/EG.09_1.pdf

[9] Microsoft: Implementing the Model-View-ViewModel Pattern [Online]. Prosinec 2013, naposledy navštíveno 05. 07. 2014.
URL http://msdn.microsoft.com/en-us/library/ff798384.aspx

[10] Microsoft: XAML Overview (WPF) [Online]. Květen 2014, naposledy navštíveno 13. 07. 2014.
URL http://msdn.microsoft.com/en-us/library/ms752059(v=vs.110).aspx

[11] Need, D.: Mitigating Airspace Issues In WPF Applications [Online]. Únor 2013, naposledy navštíveno 10. 07. 2014.
URL http://blogs.msdn.com/b/dwayneneed/archive/2013/02/26/mitigating-airspace-issues-in-wpf-applications.aspx

[12] Raskin, J.: *The Humane Interface: New Directions for Designing Interactive Systems*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, ISBN 0-201-37937-6.

[13] Reinhard, E.; Ward, G.; Pattanaik, S.; aj.: *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann publishers, Květen 2010, ISBN 978-0-12-374914-7.

[14] Shneiderman, B.; Plaisant, C.: *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. ADDISON WESLEY Publishing Company Incorporated, páté vydání, 2010, ISBN 9780321537355.
URL http://books.google.cz/books?id=2CfROgAACAAJ

[15] Solis, D.: *Illustrated WPF*. Apresspod Series, Apress, 2009, ISBN 978-143021-910-1.

[16] Zemčík, P.: Studijní opora předmětu *Tvorba uživatelských rozhraní*. Listopad 2006.
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ITU-IT/texts/lTU-Podpora.pdf?cid=9404