TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007/90 - 1 – Informační technologie

Editor videa s post-process efekty

Video editor with post-process effects

Bakalářská práce

Autor: **Stanislav Tvrzník**
Vedoucí práce: Ing. Petr Kretschmer
Konzultant: Ing. Petr Kretschmer

V Liberci 10. 5. 2012

Zde bude vloženo zadání práce

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje od mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultace s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování:

Rád bych zde poděkoval vedoucímu bakalářské práce Ing. Petru Kretschmerovi za rady a čas, který mi věnoval při řešení zadané problematiky. Poděkování patří také mé rodině, která mě podpořila a poskytla prostor pro vytvoření této práce.

Abstrakt

Tato bakalářská práce se zabývá realizací grafického editoru videa a následnému využití tohoto editoru k testování využitelnosti výkonu grafických karet při renderování efektů videa. Pro efekty je využit pixel shader model 2.0. Pixel shader je část programu napsaná pro grafické karty v prostředí DirectX. V práci je provedena analýza existujících bezplatných střihových programů a na základě této analýzy je navržen program realizující funkce, které tyto vybrané programy v základu neobsahují. Práce obsahuje také přehled použitelných technických prostředků pro realizaci načítání a ukládání videa a zároveň i jejich výhody a nevýhody.

Zpráva se zabývá funkcí samotného editoru. Editor je napsán v jazyce C++ a využívá knihovnu FFMpeg pro práci s videem. Vysvětluje principy pro načítání a ukládání videa do souboru, postupy pro zpracování jednotlivých obrazových vrstev na základě jejich typu a aplikování všech efektů na vrstvu videa. Zabývá se zejména využitím klíčových snímků pro nastavení každého efektu a přepočet hodnot v mezních pozicích pro nastavení efektu na základě definovaných klíčů. Grafické efekty jsou napsány v externích souborech, kde se dají snadno doplňovat i po kompilaci editoru. Editor obsahuje efekty nejčastěji používané při editaci filmových scén a přechodů mezi těmito scénami.

Velkou částí editoru je mimo jiné i uživatelské grafické rozhraní. To je postaveno jak z grafických prvků použitého grafického realtimového enginu Irrlicht, tak z části z grafické nadstavby nazvané jako HUD manager. Tento manager umožnuje definovat základní vzhled editoru z XML souboru pro snadnější a přehlednější změny ve vzhledu. Závěrem práce je zhodnocena především systémová náročnost a praktická použitelnost vytvořeného editoru.

Klíčová slova

Editor videa

Post-processing obrazu

C++ video kodéry

Abstract

This thesis is concerned with implementation of a graphical video editor and subsequent use of these editor to test the usability of performance graphics cards forrendering video effects. For effects is used pixel shader model 2.0. Pixel shader is part of a program written for graphics cards in DirectX environment. In this work is analyzed existing free editing programs and on the basis of this analysis is designed program implementing functions that these selected programs do not impement. The work also includes an overview of applicable technical resources to implementloading and saving the video as well as their advantages and disadvantages.

Basis of this thesis deals with the function editor itself. The editor is written in C++ and uses FFMpeg library for video processing. Explains the principles for loading and saving the video file, procedures for processing individual image layers based on their type and application of all effects on the video layer. It deals mainly with the use of keyframes to adjustment each effect and value scaling the limit positions for the effect settings based on defined keys. Graphic Effects are written in external files, which can be easily edited after compiling editor. Editor includes effects most often used for editing film clips and transitions between scenes. Graphical user interface is also a big part of the editor. It is built partially from graphic elements used in real-time graphics engine Irrlicht and partially from graphical extensions called as HUD manager. This manager allows to define the basic appearance of the editor from the XML file for easier and clearer changes in appearance. In conclusion of this work is primarily evaluated system performance and practical usability of the developed editor.

Keywords

Video editor

Picture post-processing

C++ video coders

Obsah

Seznam ilustrací

Seznam tabulek

Významy použitých zkratek

GUI – Graphics user interface
HUD – Head up display
XML – Extensible Markup Language
FPS – Frames per second, představuje rychlost vykreslování obrazu
HLSL – High level shader language
HD video – High definition video

1. Úvod

Účelem práce je ověřit využitelnost renderovacího výkonu grafických karet v osobních počítačích pro vytváření grafických efektů na již natočeném materiálu. Dále vytvořit aplikaci schopnou sloučit více navzájem se překrývajících videí do jednoho výsledného videa a realizovat několik použitelných efektů a transformací, které by bylo možné použít na každou vrstvu. Mimo to by měl podporovat práci s titulky, u kterých může uživatel měnit barvu a velikost písma.

2. Existující řešení

2.1. *Microsoft Movie Maker*

Editor videa od společnosti Microsoft je v dnešní době velmi oblíbený nástroj pro tvorbu videa. Jeho předností je velmi jednoduché ovládací rozhraní a snadná práce s videem. Program je udržován aktuální a má k dispozici doplňky, které mohou rozšířit jeho funkce. Nevýhodou je zpracování pouze jedné video a audio stopy v čase.

2.2. *Pinnacle VideoSpin*

Pinnacle VideoSpin je funkcemi velmi podobný Microsoft Movie maker. Menší rozdíl je v práci se zdrojovým videem, kdy VideoSpin ihned po otevření zdrojového videa rozdělí toto video do menších bloků podle obsahu, se kterými lze dále pracovat samostatně bez nutnosti střihu. Opět je omezen na jednu pracovní vrstvu videa v čase.

2.3. *Shrnutí*

Oba zvolené uživatelské programy patří k nejvíce používaným střihovým programům. Jsou uživatelsky přívětivé, ale značná nevýhoda je absence funkcí pro zpracování více videí ve stejném čase.

3. Kodeky pro video a audio v c++

Pro kódování videa existuje velké množství kodeků. Vytvářet program pro kódování i dekódování jednotlivých kodeků vyžaduje mnoho času k naprogramování. Z tohoto důvodu vznikly knihovny funkcí slučující jednotlivé kodeky a práci s nimi do jednotné podoby.

3.1. *Webm*

WebM je nový formát videa vytvořený k distribuci video a audio streamů po internetu. Celý formát i jeho jednotlivé části jsou vytvářeny jako open-source software. Základem je video kodek VP8, který společnost Google uvolnila pod BSD licencí po odkoupení společnosti On2 Technologies. Pro přenos zvuku se využívá kodek Vorbis. Specifikace formátu WebM umožňují uložit více audio i video streamů do jednoho kontejneru. Kontejner formátu WebM je založen na Matrjošce, který umožňuje skladovat všechny náležitosti videa v jednom souboru.

3.2. *FFMPEG*

FFMpeg je knihovna funkcí pro práci s multimédii. Poskytuje podporu pro kódování i dekódování formátů od nejstarších až po nově specifikované. Formáty, které nemají veřejné specifikace, jsou reimplementovány za využití reverzního inženýrství. Kromě schopností pro načítání multimédií je součástí knihovny i přehrávač. Knihovna je nezávislá na platformě, ale je primárně určena pro Linux. Vývojáři nevydávají často kompilované verze knihovny pro Windows, což může způsobit problémy při používání Microsoft Visual studia. FFMpeg je dostupný pod dvěma licencemi v závislosti na využití jeho jednotlivých částí.

3.3. *OpenCV*

OpenCV je knihovna funkcí vytvořená pro strojové vidění. Obsahuje tedy v základu několik efektů pro zpracování obrazu, které by bylo možné využít v editoru. Zejména velmi snadné načítání souboru s videem je předností této knihovny. Knihovna je dostupná s BSD licencí. Výhodou je jednoduché aplikační rozhraní, které umožňuje vytvořit během chvíle program pro načítání i ukládání

upraveného videa. Protože je ale funkcionalita zaměřena na strojové vidění, chybí podpora zvuku pro uložení do výsledného souboru.

4. Zpracování obrazu

4.1. *Irrlicht engine*

Irrlicht je open source realtimový renderovací 3D grafický engin napsaný v jazyce C++. Umožňuje rederovat obraz jak prostřednictvím Direct3D, tak OpenGL. Díky tomu je možné využívat aplikace s tímto enginem na více platformách současně. Předností tohoto enginu je velká komunita uživatelů doplňující do enginu nové funkce. Engin je licencován pod zlib/libpng licencí. Další informace o tomto enginu lze nalézt na domovské stránce tohoto projektu http://irrlicht.sourceforge.net/ [1].

Rendering je název techniky, při které se na základě modelu vytváří reálný obraz nebo scéna [2].

4.2. *Dekódování videa*

Pro zpracování obrazu a vytvoření efektů je nejdříve potřeba načíst správný snímek videa ze souboru. Protože ale některé formáty videa neukládají úplně snímky do souboru, je načítání snímku komplikované v případě, že je potřeba načíst snímek na jiné pozici, než je bezprostředně následující po pozici aktuální. Pro tento účel poskytuje použitá knihovna FFMpeg funkci zvanou *seek*, která přesune ukazatel pro čtení na pozici zadanou časovou značkou nebo pořadím snímku. Pokud nelze umístit kurzor přesně na požadovanou pozici, umístí se na nejbližší předchozí celý snímek. Při načítání obrazu se pak při použití funkce *seek* cyklicky načítají snímky do doby než je dosaženo zadané pozice. Načtený snímek se následně transformuje do RGB spektra a uloží do textury. Na této textuře už je možné provádět obrazové úpravy v grafickém enginu.

Pro každý otevřený soubor s videem je vytvořen samostatný dekodér. To je nezbytné vzhledem k různým formátům podporovaného videa. Protože ale jeden dekodér může být přidělen více vrstvám, mohlo by dojít k opakovanému dekódování stejného snímku a tím i předchozímu znovu-umístění kurzoru. Právě změna pozice kurzoru způsobuje značné snížení výkonu při načítání obrazu. Proto má každý kodér uloženu poslední renderovanou texturu, která zkracuje čas potřebný k načítání při opakovaném volání stejného snímku.

4.3. *Encodování videa*

Připravené video s nastavením vrstev se musí z programu uložit zpět do souboru. Výsledné video se ukládá v RAW formát, což je formát bez komprese a ztrát. Protože je program navržen jen jako mezičlánek pro realizaci speciálních efektů na krátké segmenty videa, může být další výstup bez komprese. Výsledné video se ukládá do souboru output.avi.

Aby bylo možné video bez problému použít dále, musí mít správně sestavenou hlavičku. FFMpeg má funkci pro vytvoření hlavičky a v některých případech i uzavření souboru při kódování.

5. Pracování ve vrstvách

Zpracování více rozličných zdrojů ve vrstvách je jednoduchý způsob, jak zpřehlednit uživateli práci při editaci. S každou vrstvou pracuje uživatel jako s nezávislým objektem. U každého takovéhoto objektu se nastavují efekty stejným způsobem. Proto se uživatel nemusí zajímat o to, jaký typ informace obsahuje vrstva, se kterou pracuje. V programu se všechny typy vrstvy zpracovávají stejným algoritmem. V případě rozšíření funkcí editoru v budoucnu, bude možné použít tyto nové funkce pro všechny vrstvy.

Uspořádání vrstev zároveň řeší prioritu vykreslování jednotlivých vrstev do konečného obrazu. Vrstva, která je v seznamu výše, dostane při vykreslování nižší prioritu. Vrstvy se vykreslují v pořadí od nejnižší priority k nejvyšší.

Počet současně vytvořených vrstev je programově omezen na 2000. Hlavní důvod pro toto omezení je velikost operační paměti, kterou dostane program přidělen operačním systémem. Pokud program vyčerpá přidělenou paměť, je vysoké riziko jeho pádu při pokusu o alokaci nových zdrojů. Pro snížení rizika by v budoucnu byl vhodný způsob omezit vytváření nových vrstev v závislosti na zbylé paměti programu. Pro tento krok je nejprve nutné zjistit velikost paměti potřebné k bezproblémovému chodu aplikace bez vytvořených vrstev. Zjištěná velikost paměti by následně sloužila k nastavení prahu minimální dostupné paměti.

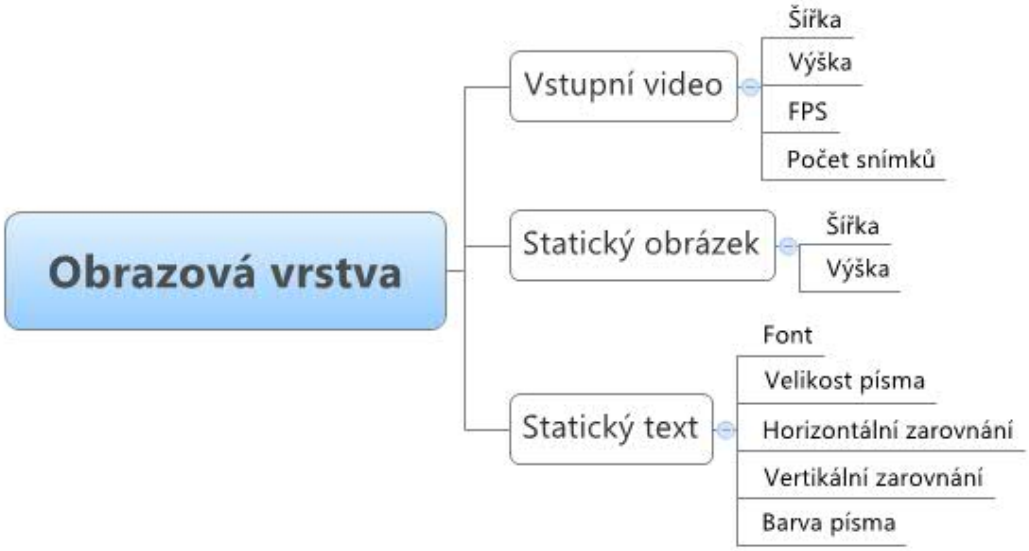
5.1. *Typy zdrojů*

V editoru lze pracovat se třemi typy zdrojů. Každý typ zdroje obsahuje jiné parametry, které definují jeho obsah. Jednotlivé zdroje videa jsou následující:

- Video
- Text
- Statický obraz

Při vytváření zdroje pro každou vrstvu se jednotlivé zdroje třídí podle přípony souboru, ze kterého se zdroj načítá. Vytváří-li se textová vrstva, vyplňuje se textový obsah vrstvy. Pro načítání obrazové vrstvy a vrstvy videa je použito dialogové okno

s výběrem souboru na disku. Výběr souboru na disku s využitím tohoto dialogu je podobný, jako při použití systémového *file dialogu*.



5.1. Ilustrace: Orientační schéma rozdělení typů zdrojů

5.2. *Zdroj videa*

Pro editor je nejpodstatnější a nejvíce využitelná právě vrstva s videem. Zdrojem této vrstvy je soubor s videm ve formátu „mp4“ nebo „webm“. Formát videa se kontroluje už při vytváření zdroje na základě přípony vybraného souboru. Tato kontrola snižuje riziko použití chybného souboru, který by mohl způsobit nestabilitu aplikace. Pro vytvoření zdroje videa zobrazí editor dialog pro výběr souboru. Výchozí složkou po prvním otevření dialogu je ta, ve které se nachází spouštěcí soubor editoru. Tato složka je zároveň pracovní složkou uchovávanou irrlicht enginem. Při procházení *file dialogu* uživatelem se mění i pracovní složka. Tohoto efektu je využito pro zachování poslední otevřené složky při vytváření více zdrojů současně, protože často bývají soubory pro jeden stejný projekt v jedné složce.

5.3. *Text*

Textový zdroj se vytváří odlišným způsobem. Pro zdroj textu jsou vyžadovány mimo vstupního textu i parametry velikosti písma a použitý font. Tyto parametry jsou vybírány z množiny sestavené při spuštění editoru.

Pracovní plocha textového zdroje vždy odpovídá velikosti vytvářeného videa. Text je uvnitř této plochy zarovnán na střed v horizontálním i vertikálním směru. Výchozí barva písma je nastavena na bílou.

5.4. *Statický obraz*

Zdroj v podobě statického obrazu se vytváří stejným způsobem, jako zdroj videa. Využívá stejný file dialog a typ zdroje, který bude z vybraného souboru vytvořen, je vybrán až podle přípony vybraného souboru. Podporována je i průhlednost obrázku.

5.5. *Podporované typy souborů pro zdroj*

Následující tabulka zobrazuje podporované typy souborů, které lze načíst při vytváření nového zdroje.

1. Tabulka: Podporované formáty souborů pro zdroje

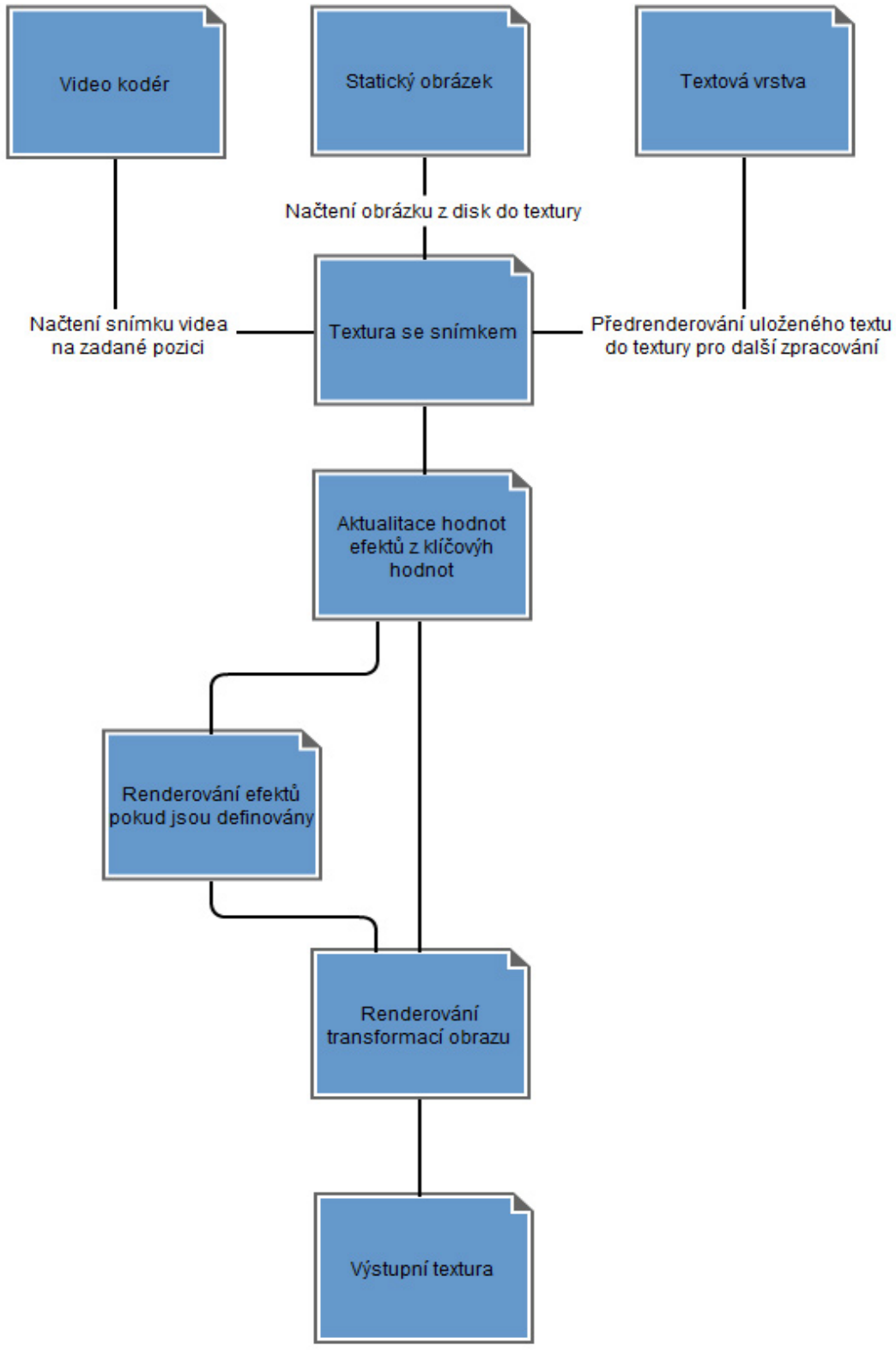
Přípona souboru	**Typ zdroje**	**Popis zdroje**
JPG	Statický obraz	Komprimovaný obrázek bez průhlednosti.
PNG	Statický obraz	Obrázek s podporou průhlednosti.
WEBM	Video	Video pro web. Kodek VP8.
MP4	Video	Video ve formátu MPEG 4

5.6. *Rizika práce se zdroji*

Při práci se zdroji v editoru nesmí dojít k přemístění nebo smazání zdrojového souboru. Editor z důvodu úspory paměti nenačítá dopředu obsah celého zdroje, ale načte pouze potřebnou část souboru v okamžiku, kdy je vykreslována.

6. Vizuální efekty

6.1. *Princip vytváření vizuálních efektů ve videu*



6.1. Ilustrace: Řetězec zpracování textury vrstvy s aplikací efektů

Pro vytváření vizuálních efektů je využíváno schopností grafické karty renderovat texturu na plochu vektorového objektu v trojrozměrném prostoru. Při vykreslování objektu do obrazu vykonává grafická karta program napsaný v shaderovém jazyce. Tento program je rozdělen do dvou částí.

6.1.1. Vertex shader program

V první části se provede transformace všech vrcholů vykreslovaného objektu. V HLSL jazyce je pojmenován jako vertex shader program. Editor využívá vertex shader pouze pro transformaci vrcholů plochy, na kterou je mapován obraz videa, do krajních pozic scény. Tím je zajištěno využití celé plochy pro obraz videa. Nároky na výpočetní výkon rostou se složitostí vykreslovaného modelu.

V případě chybějícího vertex shaderu provádí stejnou transformaci i editor při renderování. Při testech se objevili potíže se zobrazením u některých grafických karet, pokud nebyl použit vertex shader program.

6.1.2. Pixel shader program

Druhá polovina programu vykresluje povrch objektu tvořený vrcholy, takzvané polygony. Provádí transformaci na každém pixelu výstupního dvourozměrného obrazu v místech, kde se nachází vykreslovaná plocha objektu. Z toho plynou vzrůstající nároky na výkon grafické karty podle velikosti plochy, kterou objekt na scéně zabere. Náročnost výpočtů je také ovlivněna rozlišením bufferu, do kterého se renderuje scéna. Editor omezuje velikost tohoto bufferu výsledným rozlišením videa. Při pokusu o renderování většího obrazu, než je velikost bufferu dochází k oříznutí okrajů renderovaného obrazu podle nastavení efektu. Výsledkem je barva pixelu ve scéně.

Využití pixel shaderu grafické karty způsobuje další velké omezení. Neumožňuje vytvořit obrazový buffer větší, než je velikost okna editoru. Toto omezení je vinou použitého grafického enginu. Při nastavení velikosti výstupního videa větších rozměrů, než je velikost obrazovky, může dojít k nečekaným vedlejším efektům závislým na konkrétním hardware.

6.1.3. Renderování do textury

Obvykle se po vykreslení všech objektů do bufferu obsah tohoto bufferu promítne na obrazovku jako nový snímek. Na promítnutém bufferu ale není možné provádět další úpravy. Proto se místo obvyklého bufferu využívá takzvaná *Render*

target textura, do které se vykresluje scéna stejně jako do bufferu. Výhoda render target textury je možnost použít ji stejným způsobem, jako původní obraz videa a tím vytvořit nový efekt. Tento typ textur není podporován u některých starších grafických karet. Editor při každém spuštění kontroluje podporu této techniky a pokud není dostupná, automaticky se ukončí. Tím je zabráněno nechtěným pádům aplikace během práce s videem.

6.1.4. Kvalita renderovaného obrazu

Přestože jsou využity techniky pro zlepšení výsledného obrazu, které poskytuje použitý grafický engin, ztrácí obraz v některých případech na kvalitě. Při ideálních podmínkách je zachováno řádkování původního videa ve výsledném obrazu. Tohoto stavu lze dosáhnout pouze při vykreslení zdroje bez úpravy jeho nastavení. V případě změny proporcí videa je pochopitelná ztráta ostrosti obrazu. Irrlicht v tomto případě používá techniku bilineárního filtrování textury, která částečné vyhladí transformovaný obraz.

K největším ztrátám kvality dochází při opakovaném použití post-procesových efektů na renderovanou vrstvu. Obraz v tomto případě začne být rozostřený. Tento jev je vytvářen samotnou grafickou kartou, která renderuje obraz na trojrozměrný objekt a tím i automaticky opakuje filtrování mapované textury. Tímto filtrováním ztrácí s každým použitým efektem původní obraz svoji kvalitu. Potřebu filtrovaní textury nelze ale dopředu určit programově, protože některé efekty s tímto filtrováním mohou dosahovat lepších výsledků.

6.2. *Klíčové snímky*

Vytváření klíčových snímků je nástroj jak zmenšit potřebu uživatelského zásahu do nastavení měnících se parametrů efektů pro každý snímek. Bez možnosti vytvoření klíčových snímků pro nastavení parametrů efektu je potřeba ručně nastavit všechny hodnoty pro výsledný efekt u každého snímku v obraze. Je zřejmé, že náročnost takovéhoto postupu poroste se snímkovou rychlostí videa. Společně s náročností roste i nebezpečí vzniku chyby zapříčiněnou nepozorností při opakovaném nastavování podobných hodnot u sousedních snímků. Například posunutí desetinné čárky.

Při použití klíčových snímků je časová náročnost řádově nižší a případné doladění nastavení efektu se omezuje na několik málo hodnot. Klíčový snímek je možné přiřadit k libovolnému obrazovému snímku. Každý takový snímek ale může obsahovat pouze jeden klíč od každého efektu. Při použití více různých klíčových hodnot na stejném snímku by nebylo možné rozlišit výslednou hodnotu, od které se dále odvíjí hodnoty parametrů u okolních snímků.

Výpočet hodnot jednotlivých parametrů efektu mezi klíčovými snímky probíhá tak, že se naleznou nejbližší 2 klíčové snímky. První klíč musí předcházet aktuální pozici na časové ose, druhý klíč se musí nacházet za aktuální pozicí. Pokud jeden z klíčů chybí, je pro výpočet parametrů uvažován pouze existující. V případě, že existují pro danou pozici 2 klíčové snímky, je hodnota parametrů efektu u aktuálního snímku počítána lineárně podle vzorce 6.2 ze vzdálenosti mezi oběma klíčovými snímky.

Vzorec pro lineární výpočet hodnoty parametru efektu

$$hodnota = a + \frac{(b-e)*(e-c)}{d-c} \quad (6.2)$$

kde *a* je parametr klíčového snímku před aktuální pozicí, *b* je parametr klíčového snímku za aktuální pozicí, *c* je pozice předcházejícího klíčového snímku, *e* je aktuální pozice na časové ose a *d* je pozice koncového klíčového snímku. Všechny pozice snímku jsou relativní k posunu obrazové vrstvy od počátku časové osy.

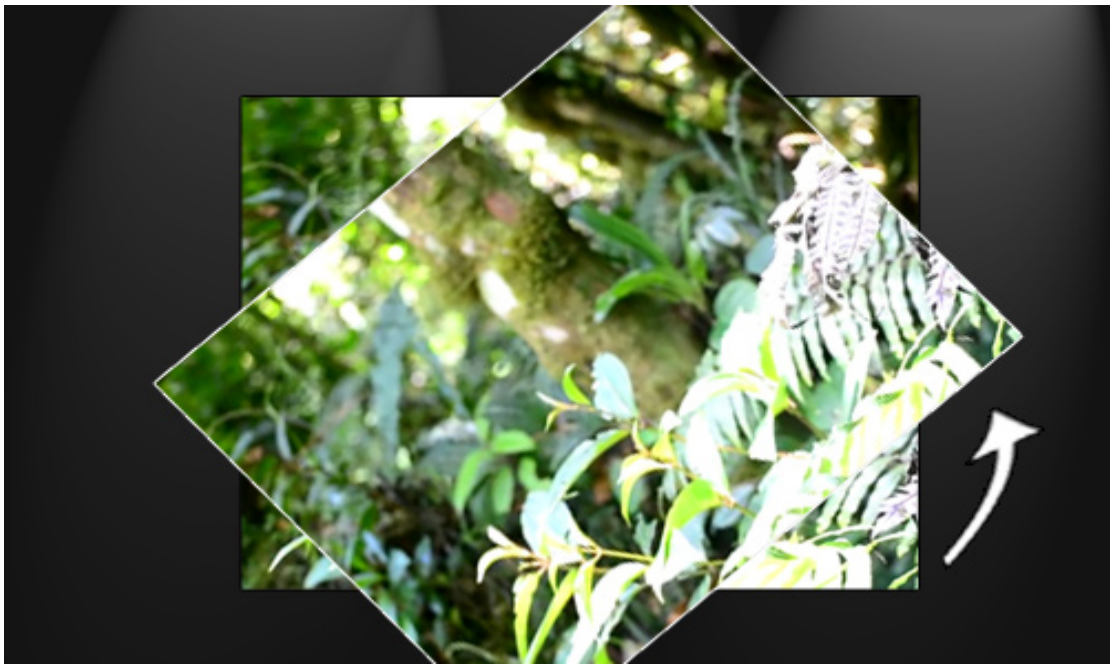
Klíčová pozice nemusí nezbytně obsahovat nastavení pro všechny efekty na dané vrstvě. Pro určení hodnoty na aktuálně zpracovávaném snímku se vždy použijí pouze existující klíče.

Pro zjednodušenou funkci klíčových snímků vytváří editor klíčový snímek s výchozím nastavením efektu na pozici prvního snímku obrazu. Pokud tedy uživatel nezadá žádné hodnoty nebo mění-li parametry efektu pro celou vrstvu, editor interně změní parametry tohoto nulového klíče a při následném zpracování obrazového efektu se postupuje stejnou metodou jako při zpracování jednoho klíče, přestože uživatel žádný klíčový snímek nevytvořil.

Při odebírání klíče ze seznamu je opět hlídán jejich počet. Pokud by se uživatel pokusil odebrat poslední klíč, editor preventivně tuto akci nepovolí, aby při vykreslování existoval výchozí parametr.

6.3. *Rotace obrazu*

Editor videa umožňuje i rotovat jednotlivé vrstvy kolem středu. Střed je určen rozměry vrstvy, která se rotuje. Pro rotaci dvourozměrného obrazu je použit starší kód z fóra Irrlicht enginu. Rotování obrazu je prováděna až na závěr zpracování všech efektů vrstvy. Tím mohou být ovlivněny některé efekty pracující s transformací v konkrétním směru. Příkladem takovéhoto efektu je například horizontální rozmazání.



6.3. Ilustrace: Efekt rotace obrazu

6.4. *Změna proporcí*

Velikost vrstvy obrazu je určena velikostí textury, která je výsledkem renderovaných efektů. Změna velikosti této vrstvy je provedena až po aplikaci všech efektů na zpracovávanou vrstvu. Dochází tak k minimálnímu zkreslení obrazu. Původní řádkování obrazu je při renderování upravené scény ztraceno. Pro zlepšení kvality transformovaného obrazu se na textury používá bilineární filtrování textur. Toto filtrování vytvoří hladké přechody mezi jednotlivými pixely textury.

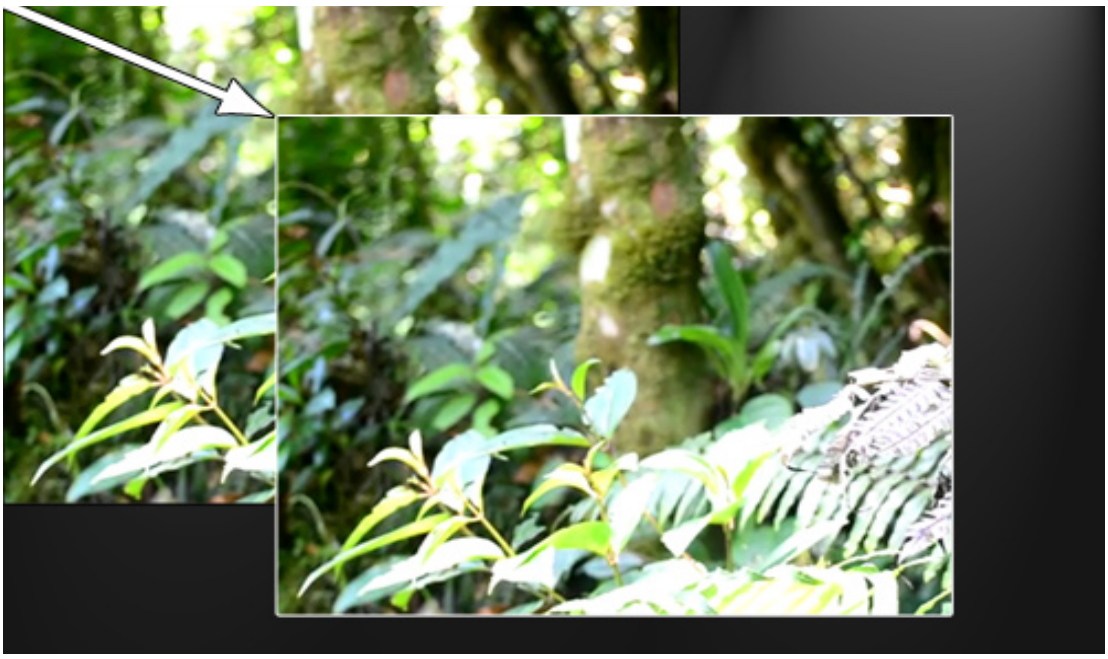
Editor pracuje s rozměry vrstvy stejným způsobem jako s efekty. Při využití klíčových snímků s různým nastavením velikosti vrstvy je mezi klíčovými snímky velikost lineárně měněna. Proporcemi je myšlena absolutní šířka a výška vrstvy v pixelech. Hodnoty obou rozměrů může nabývat pouze celých čísel. Při nižších rozlišeních videa se mohou projevit výrazné skoky velikosti.

6.4. Ilustrace: Efekt změny proporcí – výška

6.5. Ilustrace: Efekt změny proporcí – šířka

6.5. *Posun obrazu*

K posunu vrstvy dochází až při posledním vykreslení textury. Není tím ovlivněna rotace vrstvy, při které by bylo navíc nutné počítat s posunem do původní pozice. Posun vrstvy je zadáván v pixelech a hodnoty jsou omezeny na celá čísla. Bod [0,0] je v levém horním rohu videa. Souřadnice rostou do kladných hodnot ve směru osy x doprava a osy y směrem dolu. Vrstvu je možné od tohoto bodu posunout do všech směrů o velikost v rozsahu 32bitového integeru. Vykreslení posunu a změna velikosti je v editoru při zpracovávání obrazu sloučeno do jedné funkce.



6.6. Ilustrace: Efekt posun obrazu

6.6. *Efekty editoru*

Efekty editoru jsou navrženy pro snadnou manipulaci a rozšiřitelnost i po dokončení editoru. Konfigurace efektů je uložena v XML souboru. Tento soubor je umístěn v podsložce editoru s názvem effects. Úprava stávajícího nebo vytvoření nového efektu se stává ze dvou kroků.

Nejdříve se napíše nebo případně opraví soubor obsahující shader program. Shader program se píše pouze pro pixel shader a vstupní funkce musí mít název *pixelMain* s návratovou hodnotou *float4*. Tento soubor má pro jednotnou formu koncovku „fx“. Prozatím je podporován pouze HLSL kód. Dodržování této koncovky

není povinné, protože editor zpracovává pouze obsah souboru a na příponu nebere ohled.

Druhý krok při nastavování efektu je umístění jeho definice a konfigurace do XML souboru. Z tohoto XML souboru editor po spuštění vytvoří materiály. Tyto materiály jsou využity pro realizaci efektu při renderování obrazu. Následující tabulky uvádějí popis parametrů v XML konfiguraci.

2. Tabulka: Parametry XML konfigurace EffectSample

Název parametru	**Význam**
effectName	Název efektu tak, jak bude zobrazen v editoru.
effectFile	Název souboru s kódem HLSL pixel shaderu.
effectDescription	Podrobnější popis efektu. Tento parametr zatím není v programu použit.

3. Tabulka: Parametry XML konfigurace EffectParam

Název parametru	**Význam**
name	Název parametru tak, jak je uveden v shaderu.
min	Minimální hodnota, kterou může parametr nabýt.
max	Maximální hodnota, kterou může parametr nabýt.
paramDynamic	Pokud obsahuje slovo „dynamic“ zobrazí se parametr v editoru. Prázdná hodnota či jiný text zamezí změně hodnoty parametru. Toto nastavení je vhodné pro statické proměnné.

6.6.1. Ukázkový kód efektu

Ukázkový kód pro realizaci efektu chroma key.

```
sampler2D sample : register(s0);

float red = 0.0f;
float green = 0.0f;
float blue = 0.0f;
float alpha = 0.0f;
float tolerance = 0.0f;

struct VS_OUTPUT
{
       float4 Position : POSITION;   // vertex position
       float4 Diffuse : COLOR0;     // vertex diffuse color
       float2 TexCoord   : TEXCOORD0;  // tex coords
};

float4 chromaKey(float3 pixelRgb, float3 chromakeyColor,
float threshold)
{
       float3 dist = abs(pixelRgb - chromakeyColor);
```

```
        float diff = 1.0f - dot(dist, float3(1,1,1));
        float smooth = smoothstep(0, threshold, diff);
        float transit = (1.0f - smoothstep(0.0f, 0.6f, smooth));
        float alp = 1.0f - smooth;
        return float4(pixelRgb, alp * (255.0f - alpha) / 255.0f);
}

float4 pixelMain(VS_OUTPUT input) : COLOR
{
        float4 color = float4(0,0,0,0);
        color = tex2D(sample, input.TexCoord.xy);
        color = chromaKey(color.rgb, float3(red/255.0f,
        green/255.0f, blue/255.0f), tolerance/100.0f);
        return color;
}
```

6.6.2. Chroma key

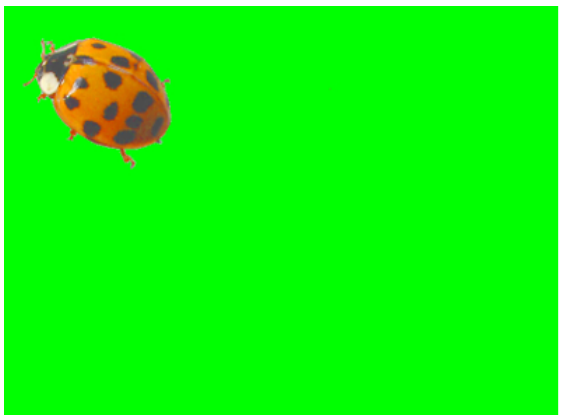
Chroma key nebo-li klíčování na barvu patří mezi často využívané efekty při vytváření videa. Používá se pro překrytí několika vrstev videa tak, že barevná plocha na pozadí jedné vrstvy se zprůhlední a zviditelní spodní vrstvu. Protože se s obrazem pracuje v RGB barevném spektru, je nejčastěji jako barva pozadí volena právě jedna ze složek tohoto spektra. Přednost se pak dává zelené a modré, protože tyto barvy mají menší pravděpodobnost výskytu.

Příkladným použitím tohoto efektu je například předpověď počasí v televizi, při kterém je promítána předpověď za osobu na obraze. Ve filmech je pak klíčování na barvu základním prvkem pro tvorbu speciálních efektů.

Postup vytváření chroma key efektu lze rozdělit na několik kroků. Nejprve se pro každý texel spočítá jeho odchylka od zadané klíčové barvy. Tato odchylka určuje výslednou průhlednost. Pokud je odchylka menší, než je tolerance, nastaví se texelu maximální průhlednost. Tímto způsobem jsou zprůhledněny velké plochy odpovídající zadané klíčové barvě. Vznikne ovšem obraz s ostrými hranami a orámovaný barvou odpovídající klíčové barvě. Pro zmírnění následků tohoto jevu je z odchylky vypočítána hodnota částečné průhlednosti pomocí HLSL funkce *smoothstep*. Tato funkce vytváří plynulý přechod v rozsahu 0 – 1, který je využit pro nastavení výsledné průhlednosti. Po tomto kroku jsou odstraněny z výsledného obrazu ostré hrany a zůstává pouze takzvaný *halo efekt*, tedy slabou záři na okrajích objektu.

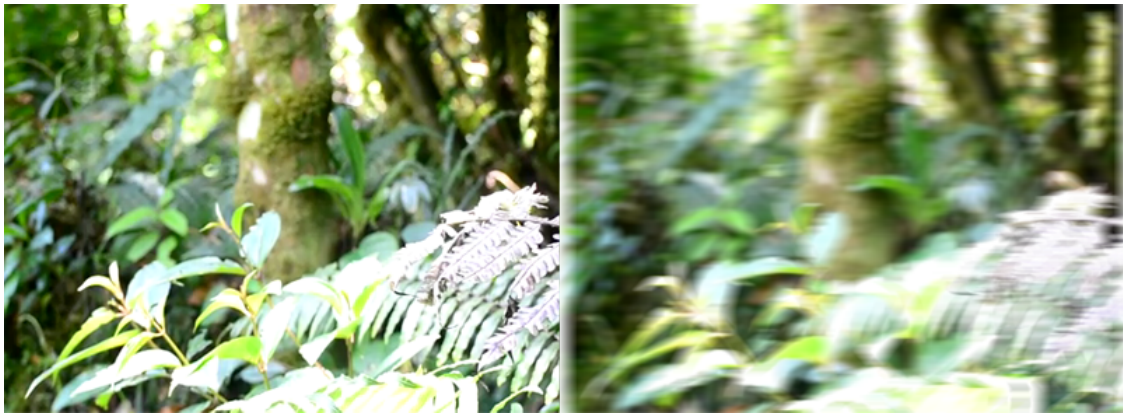
Tento efekt je možné dále vylepšit průměrovacím filtrem, který by se využil pro odečtení klíčové barvy na místech s nežádoucím efektem kolem objektu. Další

možností by bylo možné použití víceprůchodového shaderu, který by obraz před samotným klíčováním vhodně připravil.



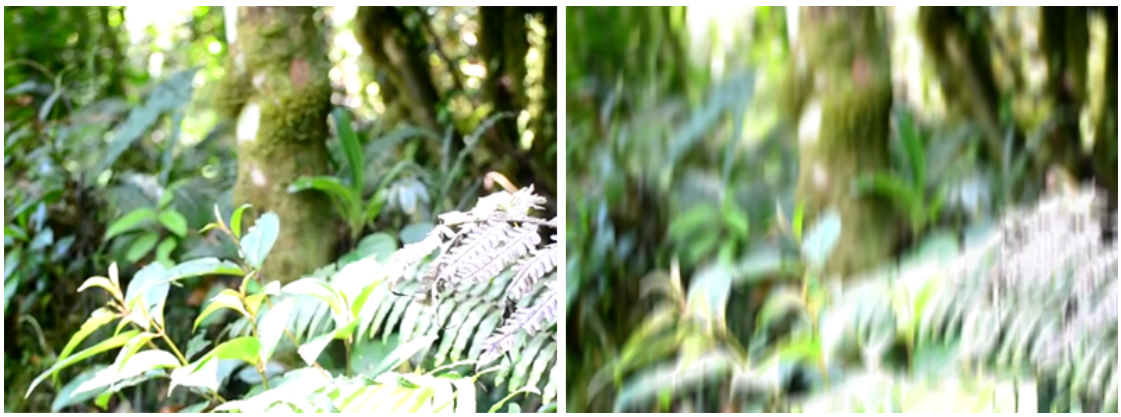
6.7. Ilustrace: Efekt zprůhlednění barvy pozadí

6.6.3. Rozmazání



6.8. Ilustrace: Rozmazání v horizontálním směru

Efekt rozmazání nebo rozostření obrazu se využívá častěji pro přechod mezi jednotlivými střihy filmu než pro speciální efekt obrazu. Při použití s vrstvením obrazových informací lze ovšem tento efekt využít i k vytvoření reálnějšímu dojmu. Příkladným použitím je vytvoření umělé hloubky ostrosti na některých vrstvách. Při několikanásobném aplikování tohoto efektu na vrstvu dochází k jeho zjemnění a zesílení zároveň.



6.9. Ilustrace: Rozmazání ve vertikálním směru

6.6.4. Sepia

Efekt sepia převádí barvy v obraze do odstínů hnědé barvy. Tento efekt je pojmenován po barvě inkoustu mořské sépie, který vypouští při útoku na kořist. Pro vytvoření sepiového efektu na obraze je využito konstantní zprůměrování barev s následným přidělením váhy hnědé barvy z RGB barevného spektra. Pro lepší výsledek je možné předřadit tomuto efektu úpravu saturace.

6.10. Ilustrace: Efekt sepiové barvy

6.6.5. Grayscale

Grayscale efekt zobrazí původní obraz v odstínech šedé barvy. Jedná se tedy o monochromatický efekt. Při jeho vytváření je vypočtena průměrná hodnota každého texelu a ta je následně přiřazena ke všem barevným složkám výsledné barvy. V nastavení efektu je navíc možné upravovat jeho intenzitu, takže lze vytvořit plynulý přechod mezi původním a černobílým obrazem podle potřeby.

6.11. Ilustrace: Efekt zobrazení ve stupních šedi

6.6.6. Průhlednost

Efekt zprůhlednění vrstvy je stejně jako rozmazání používán pro přechodové části mezi střihy. Pro nastavení průhlednosti se využívá alfa kanál textury při vykreslování. V nastavení efektu je pouze možnost měnit hodnotu průhlednosti v procentech. Zároveň je soubor se skriptem efektu možné využít jako výchozí kód pro tvorbu vlastních uživatelských efektů.

6.12. Ilustrace: Efekt zprůhlednění

6.6.7. Kontrast a saturace

Úpravou kontrastu a saturace se na natočeném videu dosahuje reálnějšího vzhledu při špatně vytvořených světelných podmínkách a přidá videu více realistický vzhled. Pro vytvoření efektu úpravy kontrastu a saturace se využívá jednoho skriptu. To umožňuje editoru změnit hodnoty kontrastu i saturace v jednom kroku.

6.13. Ilustrace: efekt zvýšení kontrastu

6.14. Ilustrace: efekt snížení kontrastu

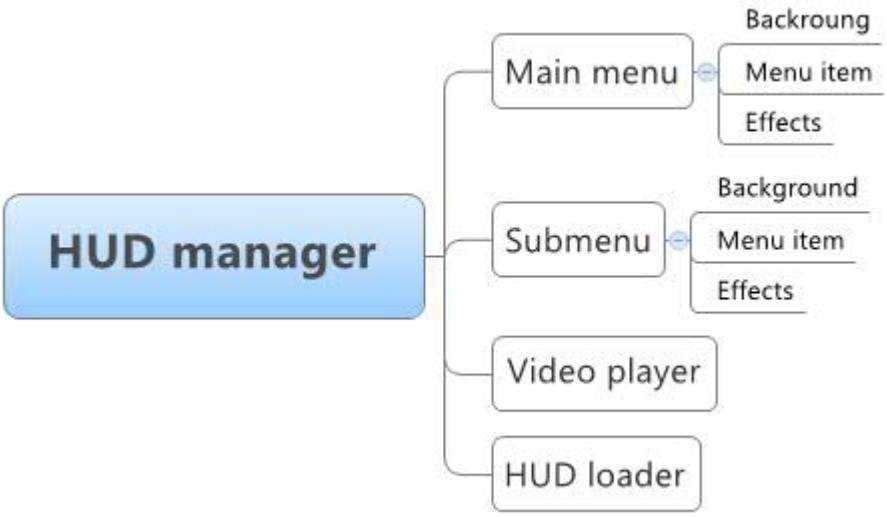
6.15. Ilustrace: efekt zvýšení saturace

6.16. Ilustrace: efekt snížení saturace

7. Uživatelské rozhraní

7.1. *GUI systém*

Celé uživatelské rozhraní je tvořené prostřednictvím GUI funkcí Irrlicht enginu. GUI zobrazuje veškeré viditelné interaktivní i statické prvky na ploše editoru. Pro část grafického rozhraní je vytvořen HUD manager. Vzhled, pozice a funkce všech prvků, které tento manager umožňuje zobrazit, je definován v XML souboru. XML konfigurace umožňuje přepracovat vzhled pracovní plochy bez opětovného kompilování zdrojového kódu editoru. HUD manager je vytvořen jako nadstavba základního GUI rozhraní.



7.1. Ilustrace: Schéma rozdělení HUD manageru

7.1.1. Popis XML parametrů HUD manageru

HUD manager je možné vhodně přizpůsobit potřebnému vzhledu konfigurací XML souboru. Konfigurační parametry uvádí následující tabulky.

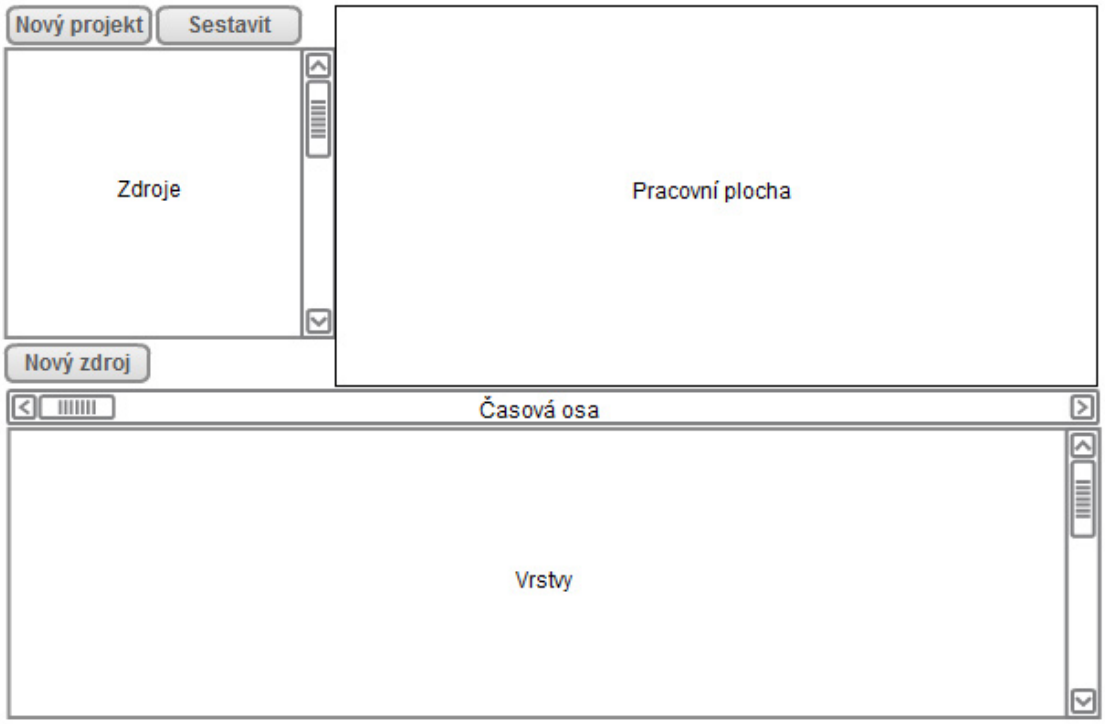
4. Tabulka: Popis parametrů XML konfigurace HUD background

Název parametru	**Popis**
index	ID menu pro aktivaci prostřednictvím událostí (hodnota musí být větší než 0).
rootID	ID kořenového menu, které musí být aktivní před zobrazením tohoto menu. Pokud není závislost, nastaví se na hodnotu 0.
texture	Textura pro pozadí menu. Není povinná (nechat parametr "").
posX	Horizontální pozice menu v okně v pixelech (měřeno od levého horního rohu).
posY	Vertikální pozice menu v okně v pixelech (měřeno od levého horního rohu).
width	Šířka menu v pixelech.
height	Výška menu v pixelech.

5. Tabulka: Popis parametrů XML konfigurace HUD objektu menu

Název parametru	**Popis**
index	ID prvku pro identifikaci události.
itemType	Typ objektu vkládaný do menu. Pokud bude vyplněn chybně, nebude objekt do menu vůbec vložen.
eventType	Určuje způsob, jakým se zpracuje událost od objektu. Možnosti jsou „core“ pro zpracování přímo v jádře programu nebo „submenu“ pro aktivaci menu podle předaného ID.
eventID	Slouží jako doplňující informace pro parametr eventType. Obsahuje ID menu, které se má zobrazit, jinak je prázdný.
name	Název elementu. Zatím není využit. Parametr může být prázdný nebo chybět.
texture	Textura pro některé typy objektů. Parametr může být prázdný.
posX	Horizontální pozice menu v okně v pixelech (měřeno od levého horního rohu).
posY	Vertikální pozice menu v okně v pixelech (měřeno od levého horního rohu).
width	Šířka menu v pixelech.
height	Výška menu v pixelech.

7.2. *Rozdělení pracovní plochy*



7.2. Ilustrace: Schéma rozvržení základní pracovní plochy

7.2.1. Menu

Hlavní menu editoru je umístěno v levém horním rohu pracovní plochy nad seznamem zdrojů. Tvoří ho pro jednoduché ovládání pouze 2 tlačítka. První tlačítko otevírá okno pro nastavení nového projektu. Druhé tlačítko spustí sestavení vytvořeného videa do výstupního souboru. Po aktivaci procesu sestavení se začnou renderovat jednotlivé snímky videa, které se zobrazují v náhledovém okně a zároveň se ukládají do výstupního souboru s názvem „output.avi“.

7.2.2. Zdroje projektu

Zdroje pro projekt jsou zobrazeny v listboxu a jsou programově omezeny na tisíc položek. Jména zdrojů jsou odvozeny od typu zdroje. Pokud je zdroj vytvořen ze souboru, tedy obrázek nebo video, odpovídá název zdroje názvu souboru. Pokud je zdrojem vytvořený text, je název zdroje vytvořen z prvních čtyřicet písmen tohoto textu.

Na řádku každého zdroje jsou dvě tlačítka se znaménky „+“ a „-“. Tyto tlačítka se používají k operacím se zdrojem na stejném řádku. Tlačítko „+“ vytváří ze zdroje novou vrstvu ve videu. Nové přidání vrstvy vynutí překreslení aktuálního obrazu, takže uživatel ihned vidí, jestli je přidaná vrstva z požadovaného zdroje. Nová vzniklá vrstva se umístí na konec seznamu vrstev a dostane nejvyšší prioritu vykreslení. Překryje tedy všechny předchozí vrstvy, protože se renderuje jako poslední. Tlačítkem „-“ se naopak odebírá zdroj ze seznamu zdrojů. Odebrání zdroje způsobí i odebrání všech vytvořených vrstev, které tento zdroj využívají.

Ve spodní části seznamu zdrojů je umístěno tlačítko pro vytvoření nového zdroje. Toto tlačítko otevírá nové okno pro přidání nového zdroje rozdělené do dvou záložek. První záložka obsahuje file dialog pro výběr souboru na disku. Výběrem souboru se vytváří zdroje typu video a statický obrázek. Druhá záložka zobrazuje nastavení pro vytvoření textového zdroje. Textový zdroj vyžaduje pro vytvoření text, který bude zobrazen, velikost písma a jeho typ. Po otevření souboru nebo kliknutí na tlačítko *vytvořit vrstvu* se okno pro vytvoření nového zdroje zavře.

7.2.3. Vrstvy a časová osa

Přehled aktivních vrstev a interpretace časové osy je umístěna ve spodní části pracovní plochy. Časová osa je tvořena posuvníkem s hodnotou od nuly do maximálního počtu snímků aktuálně vytvořeného projektu. Funkce posuvníku představující při editaci časovou osu se změní na orientační zobrazení postupu. Polohu posuvníku lze uživatelsky dále měnit, ale jakákoliv změna je ignorována až do dokončení sestavení výstupního videa.

Seznam vrstev je tvořen obdobným způsobem, jako seznam zdrojů. Každá vrstva má svůj název vycházející z názvu zdroje, ze kterého je vrstva vytvořena. Na každém řádku jsou mimo názvu vrstvy umístěna čtyři tlačítka. Tlačítko odstranění vrstvy odebere ze seznamu vrstvu na stejném řádku i s nastavením. Protože pořadí vrstev určuje jejich prioritu při renderování a překryv, jsou další dvě tlačítka určena ke změně pořadí vrstvy. Umožňují posun vrstvy o jednu úroveň v obou směrech, pokud se nejedná o vrstvu na začátku nebo konci seznamu. Poslední tlačítko otevírá okno s nastavením vrstvy.

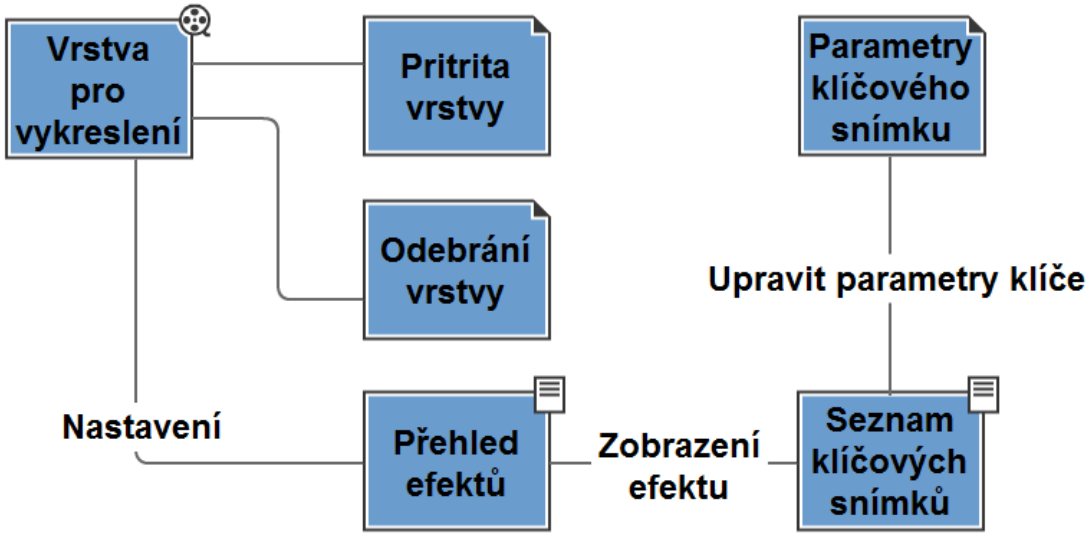
7.2.4. Náhled videa

Plocha s náhledem videa je umístěna v pravé horní části pracovní plochy a zabírá největší část vyhrazeného místa. Tvořena je dvěma statickými obrazovými plochami. První tvoří podklad pracovní plochy a je použit pouze pro grafické účely. Druhá plocha zobrazuje náhled posledního vyrenderovaného snímku výsledného videa. Na této pracovní ploše není vytvořeno žádné uživatelsky interaktivní prostředí, což tvoří velmi vhodný prostor pro budoucí rozšíření.

Rozměry obrazové plochy se přizpůsobují rozměru výsledného videa podle konfigurace projektu. Pokud je výstupní video větší než pracovní plocha náhledu (580 pixelů šířka a 400 pixelů výška), využije tato obrazová plocha maximální možnou velikost. Obrazová textura v této ploše se ale bude v takovémto případě zobrazovat zkresleně. V případě, že je obrazová plocha menší než maximální možná velikost, zobrazí se v přesném rozlišení a je zarovnána na střed.

7.3. *Ostatní menu*

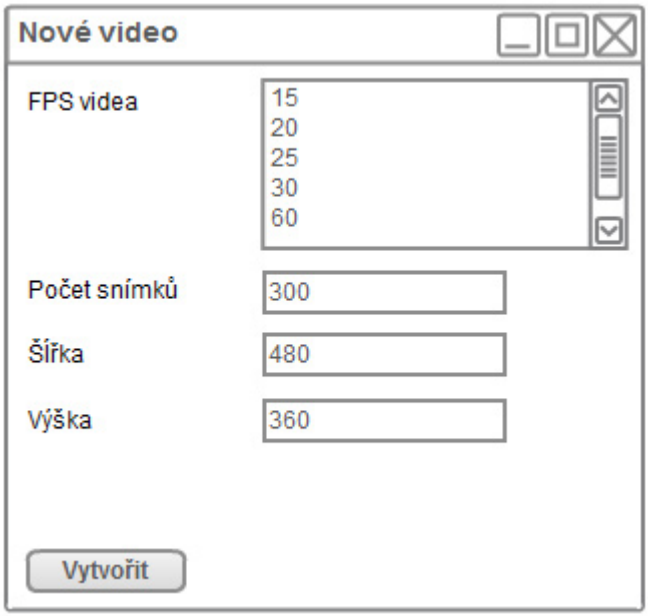
Všechna ostatní menu se uživateli zobrazují v plovoucích oknech. Tato okna může uživatel přesouvat tažením za záhlaví s titulkem okna v případě, že by překážela v zobrazení informací skrytích pod nimi. Zároveň je dodržováno pravidlo, aby se ze dvou navzájem souvisejících oken zobrazovalo vždy jen to, které je ve struktuře nejvíce podřazené. Pokud je například otevřeno okno se seznamem efektů vrstvy, a následně dojde z tohoto otevřeného okna k zobrazí okna podřazeného, bude po dobu zobrazení tohoto podřazeného okna původní okno skryto.



7.3. Ilustrace: Propojení kaskády grafických oken editoru pro nastavení vrstvy

7.3.1. Nastavení nového projektu

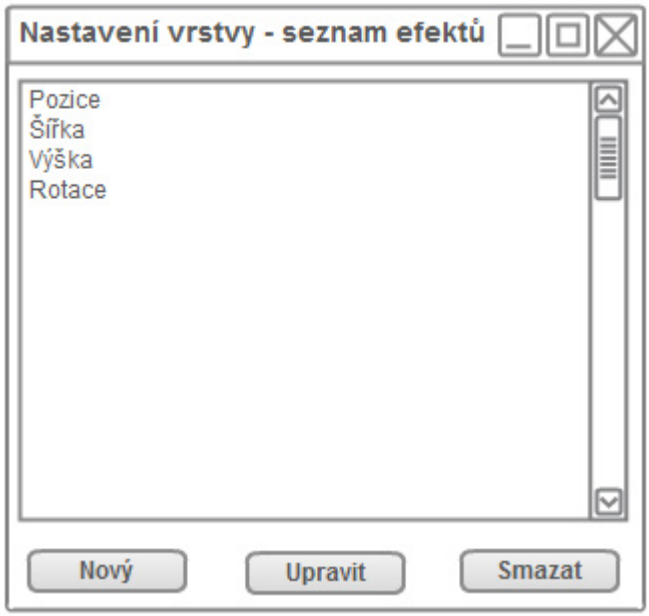
Okno pro nastavení nového projektu umožňuje znovu definovat rozměry, fps a délku sestavovaného videa. Tento krok je potřeba provést před začátkem editování videa, protože dochází k nové inicializaci prostředí. Všechny dříve existující zdroje jsou odstraněny.



7.4. Ilustrace: Okno pro vytvoření nového videa

7.3.2. *Nastavení vrstvy videa*

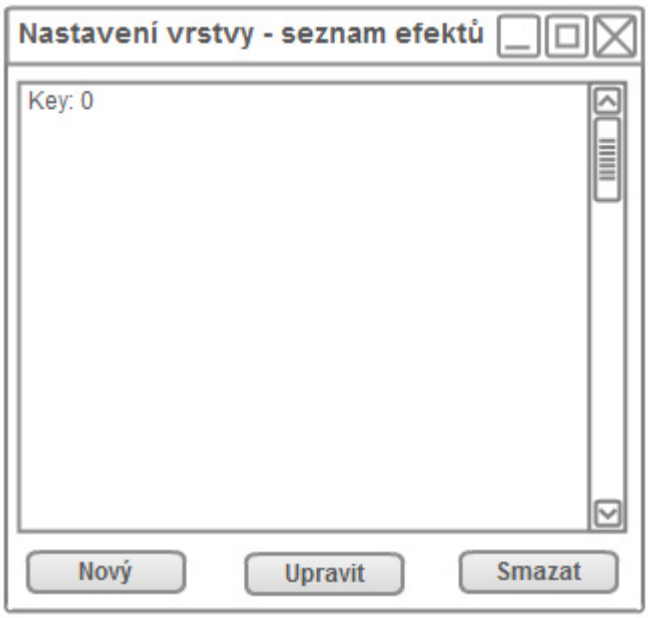
Okno s nastavením vrstvy videa zobrazuje přehled všech existujících efektů na vybrané vrstvě a funkce pro základní editaci těchto efektů. Pro editaci jsou v dolní části okna umístěny tři tlačítka. Tlačítko pro vytvoření nového efektu zobrazí nové okno s výběrem dostupných efektů. Další tlačítko *upravit efekt* zobrazí okno s nastavením efektu, který je zvýrazněn v seznamu efektů vrstvy. Poslední tlačítko vymaže zvýrazněný efekt ze seznamu, pokud se jedná o post-proces efekt.



7.5. Ilustrace: Okno zobrazující nastvení vrstvy

7.3.3. *Nastavení efektu vrstvy*

Okno zobrazující nastavení vybraného efektu vrstvy je velmi podobné předchozímu oknu s nastavením vrstvy. Rozdíl je pouze v zobrazených informacích v seznamu. Na místě, kde byly v okně nastavení vrstvy zobrazeny jednotlivé efekty na vybrané vrstvě, zobrazuje nastavení efektu všechny existující klíče, které efekt používá.

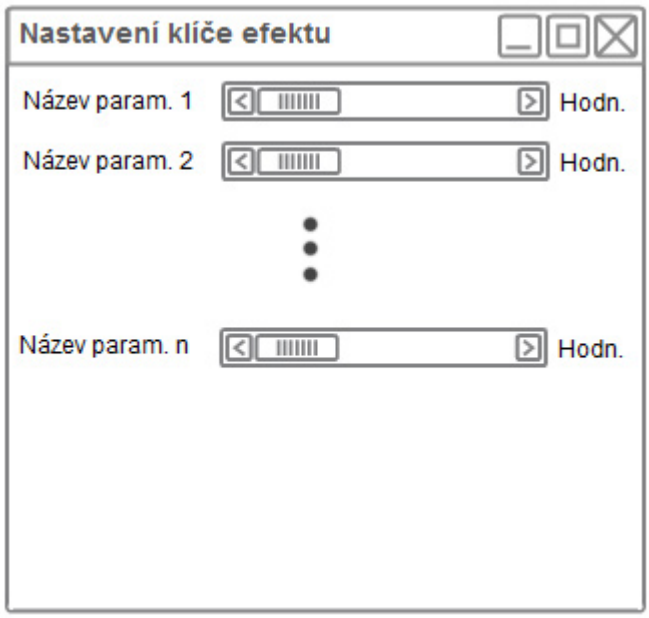


7.6. Ilustrace: Okno zobrazující nastavení efektu

7.3.4. Nastavení klíčového snímku efektu

Okno s nastavením vybraného klíče efektu je posledním oknem v kaskádě při editování vrstvy. Na rozdíl od předchozích oken, při nastavení vybraného klíče se zobrazené prvky pro editaci mění podle efektu. Každý efekt má jiný počet, název a rozsah jednotlivých parametrů. Zachování této variability je dosaženo uchováváním prototypu konfigurace klíče každého efektu, podle kterého se sestavuje grafické menu. Zároveň je z prototypu možné načíst výchozí hodnoty parametrů při vytváření nového klíče.

Protože je počet grafických prvků v tomto menu odlišný, je problematické synchronizovat popisky obsahující hodnotu parametru s aktuální hodnotou posuvníku nastavující novou hodnotu parametru. Tento problém je vyřešen přidělením stejného identifikátoru všem posuvníkům v menu. Při změně hodnoty libovolného posuvníku se aktualizují všechny hodnoty současně.



7.7. Ilustrace: Okno zobrazující rozvržení ovládacích prvků parametru klíče

8. Závěr

Editor splňuje všechny zadané funkce. Umožňuje zpracovávat více videí v jednom časovém okamžiku ve stejném obraze za použití vrstev a poskytuje pro tyto vrstvy několik praktických efektů. S využitím externích C++ knihoven bylo v editoru naprogramováno rozšířené grafické prostředí editovatelné v XML souboru. Toto prostředí umožňuje používat textury pro pozadí a animovaná tlačítka. Nebylo ale použito na celý editor, protože je problematické a nepřehledné svázat GUI eventy enginu s tímto grafickým rozšířením. Ostatní GUI prvky jsou součástí grafického enginu.

Nejrozsáhlejší částí vytvořeného programu je systém vrstev a klíčových snímků. Tato část je úzce provázána s grafickým prostředím a zajišťuje renderování snímků ve správném pořadí a s nastavenými efekty. Je postavena pouze na grafickém enginu Irrlicht bez dalších externích zdrojů.

Zbylé části vytvořeného programu jsou určeny k ukládání a načítání videa, vytváření textur jednotlivých vrstev pro další zpracování a zpracování TrueType fontů.

Mimo vlastní kód editoru jsou k editoru vytvořeny i XML konfigurace pro grafické prostředí a shaderové efekty. Součástí konfigurace efektů jsou i vytvořené HLSL efekty v samostatných souborech. Tyto zdroje jsou uloženy v samostatných souborech, aby je bylo možné upravovat a rozšiřovat i po sestavení programu.

Pro texty v editoru je použita knihovna z Free type projektu. Tato knihovna rozšiřuje podporu fontů v editoru. Díky tomu editor využívá všechny True type fonty nainstalované v operačním systému Windows. Nastavení velikosti textu je při jeho vytváření také umožněno.

Zvolený grafický engin disponuje pouze omezeným grafickým uživatelským rozhraním. Proto je ovládání editoru navrženo poněkud nestandardním stylem. Větší nevýhodou při editování videa je nemožnost přizpůsobit velikost okna aplikace. Pro grafické prostředí by bylo vhodnější použít platformu .NET.

Při praktickém použití se projevily větší výkonové nároky na grafické karty. Protože editor využívá stejné prostředí pro vykreslování efektů vrstev i grafického rozhraní editoru, dochází při větším množství vrstev k velkému poklesu FPS editoru.

Z tohoto důvodu se nejeví použití renderovacího výkonu grafické karty s využitím shaderových technik pro zpracování videa jako dobrá volba.

Při závěrečném testování aplikace se projevilo nezvyklé chování při vytváření nových klíčových snímků, kdy je v některých případech postup pro vytváření klíče provést dvakrát, aby se podle klíče aktualizoval efekt. Důvod tohoto problému se nepodařilo zjistit.

8.1. *Využití systémových prostředků*

Editor pro start potřebuje kolem 30 MB operační paměti. Tato nízká spotřeba paměti umožňuje velmi rychlý start aplikace. Další paměť je následně za běhu spotřebovávána zejména při vytváření nových zdrojů. Potřebná paměť pro vytvoření jednoho zdroje se pohybuje v rozmezí 3 MB až 8 MB v závislosti na velikosti výstupního videa. Přesná velikost opět závisí na rozlišení výstupního videa. Při průměrném pracovním počtu 30 zdrojů je i nadále potřebná paměť hluboko pod limitem dnešních operačních systémů. Rychlost renderování je ovlivněna zejména grafickou kartou. Nejvíce paměti editor vyžaduje při procesu sestavení videa a jeho uložení do souboru, protože dočasně alokuje několik pracovních textur.

6. Tabulka: Orientační hodnoty potřebné paměti na zdroj podle rozlišení videa

Rozlišení obrazu	**Orientační hodnota paměťi na zdroj**
480x360 pixelů	0,9 MB
800x600 pixelů	1,4 MB
4000x2600 pixelů	10,8 MB

8.2. *Hardwarové a softwarové limity*

Aplikace pracuje s pevně stanovenou výškou 800 pixelů a šířkou 600 pixelů, což je v době vytvoření aplikace dostatečně nízké rozlišení na bezproblémové použití. Pro spuštění a renderování efektů je vyžadována podpora pixel shaderu 2.0 u grafické karty. Největším omezením je ale velikost grafického bufferu, do kterého je renderován obraz. Použitý grafický engin neumožňuje renderovat obraz větší než je velikost okna, což znemožňuje použití editoru pro HD video. Přitom HD video je v době vytvoření tohoto editoru již velmi častá součást moderních editačních programů. Pro irrlicht existuje patch, který tuto podporu doplňuje. Není ale zaručeno,

že by nezpůsobil nestabilitu aplikace. Volná pracovní paměť by měla být alespoň 256 MB.

Seznam použité literatury

IZAK. Wma decoding with ffmpeg. *Stackowerflow* [online]. 21. 12. 2010. 2010 [cit. 2012-05-09]. Dostupné z: http://stackoverflow.com/questions/4487263/wma-decoding-with-ffmpeg

BOHME, Martin. Tutorial 01: Making Screencaps. *An ffmpeg and SDL Tutorial* [online]. 2003 [cit. 2012-05-09]. Dostupné z: http://dranger.com/ffmpeg/tutorial01.html

BELLARD, Fabrice. Libavcodec/api-example.c. *FFmpeg Documentation* [online]. 2001, 9. 5. 2012 [cit. 2012-05-09]. Dostupné z: http://cekirdek.pardus.org.tr/~ismail/ffmpeg-docs/api-example_8c-source.html

THEADMIRAL. Chroma Keyer/Green Screen. *Gamedev.net* [online]. 2007 [cit. 2012-05-09]. Dostupné z: http://www.gamedev.net/topic/471433-chroma-keyergreen-screen/

NEWMAN, Garry. HLSL Postprocessing pixel shader tutorial. *Facewound* [online]. 2003 [cit. 2012-05-09]. Dostupné z: http://www.facewound.com/tutorials/shader1/

STRATTON, Cort. Revel, the Really Easy Video Encoding Library. *Sourceforge* [online]. 2004, 12. 9. 2004 [cit. 2012-05-09]. Dostupné z: http://revel.sourceforge.net/

DABLER, A0, PASTORIBOT, Milan KERŠLÁGER, RAGIMIRI, HERAKLEITOSZEFESU a GANIMOTH. FFMpeg. *Wikipedie* [online]. 2007, 14. 1. 2012 [cit. 2012-05-09]. Dostupné z: http://cs.wikipedia.org/wiki/FFmpeg

About FFmpeg. *FFMpeg* [online]. 2007 [cit. 2012-05-09]. Dostupné z: http://ffmpeg.org/about.html

[1] HYBRID. *Irrlicht engine*: *A free open source 3D engine* [online]. 2011, 8. 5. 2012 [cit. 2012-05-10]. Dostupné z: http://irrlicht.sourceforge.net/

The WebM Project: *an open web media project* [online]. Copyright 2010 - 2012 [cit. 2012-05-14]. Dostupné z: http://www.webmproject.org/

The FreeType Project [online]. 4. 12. 2010 [cit. 2012-05-14]. Dostupné z: http://www.freetype.org/

[2] JUANDEV, SLADY, BOTA47, PORTHOS, ZACATECNIK, Milan KERŠLÁGER, Arthur MURRAY a LOUPEZNIK. Renderování. *Wikipedie*: *Otevřená enciklopedie* [online]. 13. 1. 2006, 10. 4. 2012 [cit. 2012-05-16]. Dostupné z: http://cs.wikipedia.org/wiki/Renderov%C3%A1n%C3%AD