VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

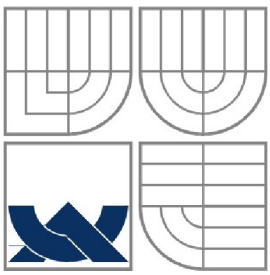
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NÁSTROJ PRE STRIH VIDEO SEKVENCIÍ

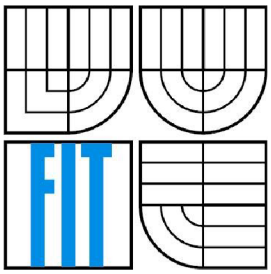
BAKALÁRSKA PRÁCA
BACHELOR´S THESIS

AUTOR PRÁCE MARIÁN KRIVDA
AUTHOR

BRNO 2007



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

NÁSTROJ PRE STRIH VIDEO SEKVENCIÍ

VIDEO EDITING CORE

BAKALÁRSKA PRÁCA
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MARIÁN KRIVDA

VEDÚCI PRÁCE
SUPERVISOR

ING. SUMEC STANISLAV, PH.D.

BRNO 2007

Abstrakt

Táto práca sa zaoberá návrhom a implementáciou jadra pre strih video sekvencií. To v sebe zahŕňa tri fázy. Prvou fázou je skúmanie dostupných nástrojov umožňujúcich strih videa. Druhou je návrh architektúry strihacieho jadra a možností jeho rozšírenia. Poslednou fázou je samotná implementácia jadra jeho rozšírení.

Kľúčové slová

video sekvencia, strih videa, spracovanie videa

Abstract

This work consist of the design and implementation of the video editing core. It contains three general phases. The first of them is inspection of video editing tools. The video editing core design is second phase. Last phase is implementation.

Keywords

video sequence, video editing, video processing

Citácia

Marián Krivda: Nástroj pre strih video sekvencií, bakalárska práca, Brno, FIT VUT v Brne, 2007

Nástroj pre strih video sekvencií

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Stanislava Sumca, Ph.D.
Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Marián Krivda
7.5.2007

Poďakovanie

Na tomto mieste by som rád poďakoval Ing. Stanislavovi Sumcovi, Ph.D. za jeho odbornú pomoc pri spracovaní tejto práce.

Obsah

1 Úvod

Spracovanie (strih a úprava) video záznamov je v súčasnej dobe veľmi rozšírené, preto dnes už nepatrí medzi výsadu profesionálov (profesionálnych filmových štúdií, ...). Klesnutím cien video techniky a rozšírením počítačov do domácností sa úprava a spracovanie videa stala zaujímavou pre mnoho amatérov. Po zlacnení video techniky a rozšírení počítačov bola potrebná ešte jedna vec, ktorá umožnila spracovanie videa na bežnom počítači a to softvér na úpravu a strih videa. Programov na prácu s videom existuje mnoho (či už komerčných alebo nekomerčných), každý je zameraný na inú oblasť spracovania, každý vyžaduje rôznu úroveň znalostí, jedna skupina je zameraná skôr amatérov, druhá patrí výhradne do rúk profesionálov, avšak všetky všetky majú jeden spoločný znak, a to že všetky pracujú s digitálnym videom. Táto práca skúma možnosti súčasných programov a knižníc na prácu s digitálnym videom. Na základe tohto poznania navrhuje architektúru a implementáciu jadra na strih video sekvencií. V tejto práci sa venujeme iba strihu digitálneho videa, ale digitálna forma nie je jediná forma, v ktorej video existuje, dokonca ani nebola prvá. Počiatky filmovania siahajú až do roku 1893, kedy Thomas Edison predstavil prvú kameru. V tejto dobe sa video zaznamenávalo výlučne na analógové médiá a počítače neexistovali.

Histórii a rôznym formám strihu sa práve venuje prvá kapitola. Nasledujúca kapitola skúma možnosti súčasných programov na prácu s video záznamami. Zameraná je jednak na komerčné nástroje, avšak nesmieme zabudnúť na nástroje nekomerčné, po väčšine opensource, ktoré sú skôr vhodné pre amatérov. V silách tejto práce nie je možné opísať všetky dostupné programy, preto boli vybratí jednotliví zástupcovia z obidvoch kategórií. Tretia kapitola preberá problematiku ukladania video súborov. Čiastočne sa tu spomína kompresia audia/videa, ďalej sa venuje rôznym obrazovým formátom (RGB, YUV, ...) a vysvetľuje ich princíp. Nepovšimnuté neostali ani kodéry/dekodéry a kontajnery video súborov (AVI, MPEG). Obsah piatej kapitoly tvorí popis knižníc na prácu s video súbormi. Doteraz bolo vymyslených mnoho kodérov/dekodérov (kodek - **ko**dér/**dek**odér), z ktorých každý má svoje výhody a nevýhody, každý je určený pre iný typ videa, … Tieto knižnice práve zjednodušujú a zjednocujú prístup k týmto kodekom, čím výrazne uľahčujú programátorom prácu. Obsahovo najrozsiahlejšia šiesta kapitola nás prevedie návrhom a implementáciou strihacieho jadra, detailne opisuje jeho architektúru, zamýšľa sa nad jeho prednosťami aj nedostatkami. Kedže toto jadro je rozšíriteľné, táto kapitola ukazuje správne postupy jeho rozširovania.

2 História digitálneho strihu

Pri strihu videa sa jedná o spojenie, rozdelenie, skrátenie, izoláciu, odstránenie alebo prechod nejakej video sekvencie za prípadného doprovodu zvuku. Či sa jedná o film, rodinné video, hudobný klip, … ani jeden sa neobíde bez úpravy, ktorú nazývame strih. Bez nej by jeho ďalšia prezentácia nemala žiadny zmysel.

Strih môžeme rozdeliť podľa rôznych kategórií napr. na:

- analógový a digitálny strih
- lineárny strih a nelineárny strih

2.1 Analógový a digitálny strih

2.1.1 Analógový strih

S analógovým strihom sa v dnešnej dobe už takmer nestretneme. Existovali analógové strihacie pulty alebo drahé videorekordéry, ktoré umožňovali strih buď za použitia vnútornej pamäte alebo s použitím dvoch pások. V súčastnosti používame analógové zdroje videa, ktoré je väčšinou lepšie previesť na digitálne a v tejto digitálnej podobe s nim pracovať ďalej.

2.1.2 Digitálny strih

Digitálny strih priniesol množstvo výhod oproti analógovému strihu. Či sa jedná o časovú náročnosť, alebo o náročnosť na ovládanú techniku. Jeho kvalita závisí na dokonalosti použitého hardvéru a softvéru. Digitálny strih sprístupnil úpravu videa širokej verejnosti svojimi neveľkými požiadavkami na použitú techniku (s požiadavkami na kvalitu rastú aj nároky na techniku, ale na obyčajný strih stačí kamera a počítač, pri tomto postupe sa analógové video prevedie do digitálnej formy, s ktorou sa pokračuje ďalej v práci). Veľkou výhodou digitálneho strihu je spracovanie na počítači, ktoré otvára nové možnosti pri jeho spracovaní (umožňuje aplikovať rôzne video efekty, animácie, prelínania, ...), ktoré sú nedostupné pri analógovom strihu. Taktiež digitálna forma videa je výhodná pre archiváciu videa a jednoduchú manipuláciu s ním, video je uložené na pevnom disku (diskovom poli) a ak s ním chcem pracovať len ho v danom programe otvorím, naproti tomu analógové video sa archivuje na páskach, s ktorými sa manipuluje mechanicky čím sa opotrebúvajú a strácajú na kvalite..

2.2 Lineárny a nelineárny strih

2.2.1 Lineárny strih videa

Je proces výberu, usporiadania a modifikovania obrazu a zvuku nahraných na videopáske alebo filme. Predtým než vznikol nelineárny strih video, bol pre lineárny strih videa zaužívaný názov strih videa.

Pôvodne sa video páska upravovala fyzicky rozrezaním a opätovným spojením (zalepením). Stopy by mali byť prerušene v mieste vertikálnej synchronizácie, bez rušenia párneho/nepárneho poradia snímku (jeden snímok je tvorený z dvoch pol-snímkov). Pretože čítacie audio a video hlavy boli od seba vzdialené niekoľko centimetrov, nie je možné spraviť súčasne korektný audio a video strih, takže sa strih vykoná najprv na video stopách, následne je audio stopa skopírovaná na správne miesto. Použitie takéhoto prístupu je veľmi obtiažne a vyžaduje veľa času, pevné nervy a jemný cit.

Lepším riešením je použitie dvoch video rekordérov, kde sa z jednej pásky kopíruje na druhú. Proces je lineárny, pretože na finálnej páske musia byť klipy kopírované v postupnom poradí (jeden po druhom), akonáhle je klip nakopírovaný na páske, nieje možné pred neho umiestniť ďalší klip bez toho aby sme zmazali to čo tam je nahraté. Je absolútne nevyhnutné nový materiál nakopírovať na novú pásku a k nemu pridať starý materiál. Každé kopírovanie prináša zo sebou degradáciu obrazu a zvuku (digitálny strih úpravami nestráca na na svojej kvalite) čo je veľmi nežiadúce.

2.2.2 Nelineárny strih videa

Nelineárny strih je moderná metóda úpravy videa, ktorá umožňuje prístup ku každému jednému snímku v hociktorom okamžiku. Táto metóda strihu je podobná konceptu „rež a spoj“ používaného pri filme pred nástupom digitálneho strihu, avšak keď sa pracuje s filmom je to deštruktívny proces. Nelineárny digitálny strih je nedeštruktívny proces uprav video sekvencií.

Video a audio dáta sa najprv prevedú do digitálnej podoby na pevný disk alebo iné úložné médium (CD, DVD, diskové polia, …). Pri nelineárnom strihu videa nie sú zdrojové súbory modifikované alebo inak menené (nemení sa ich kvalita). Väčšina programov si zaznamenáva všetky úpravy do pamäte (EDL - Edit decision list, strihová súpiska), preto je veľmi jednoduché tieto úpravy odvolať alebo ich zmeniť (zmenou EDL). Strate kvality sa vyhneme tým, že strihací softvér interne pracuje s nekomprimovaným videom (vstupné a výstupné video sekvencie môžu byť komprimované, no vo vnútri programu sa pracuje s nekomprimovaným audiom/videom).

V porovnaní s lineárnym strihom, nelineárny strih disponuje flexibilitou podobnou strihu filmu, ale s jednoduchou projektovou organizáciou. S pomocou EDL môžeme upraviť video v nízkom rozlíšení a finálny strih vykonať vo vysokom rozlíšení s použitím uloženého EDL. Takto je možné pracovať s vysokou kvalitou videa na bežných osobných počítačoch nedisponujúcich dostatočným výpočetným výkonom na prácu s videom vo vysokej kvalite v reálnom čase.

3 Nástroje na digitálny strih

Táto kapitola postupne predstavuje nástroje používané na spracovanie video sekvencií. Spracovanie videa neznamená len umiestnenie (príp. skrátenie, aplikovanie filtrov, ...) video sekvencií na správne miesto vo finálnom zázname, na čo slúžia predovšetkým komerčné strihacie štúdiá. Do spracovania videa patrí aj napr. zmena kodeku (zmena vzorkovacej frekvencie, …), v ktorom je sekvencia (či už video alebo audio) vytvorená. Na podobné účely si vystačíme aj s malými utilitami, ktoré sú väčšinou zadarmo, a ktorých ovládanie je jednoduchšie ako ovládanie kompletného strihacieho štúdia.

3.1 Komerčné nástroje – Sony Vegas

Sony Vegas je nelineárny video editor, za ktorým stoji firma Sony Media Software. Momentálne beží iba na operačných systémoch MS Windows 2000/XP/2003/Vista. Pôvodne bol vyvíjaný firmou Sonic Foundry, od verzie číslo 4 ho vyvíja spoločnosť Sony, po tom čo odkúpila časť spoločnosti Sonic Foundry. Svoj život začínal ako čistý audio editor s čiastočným zameraním na prevzorkovanie audia, čo z neho spravilo lídra v oblasti nelineárnych video editorov s omnoho prepracovanejšou podporou práce s audiom oproti ostatným nelineárnym editorom.

Vegas bol prvým nelineárnym editorom poskytujúcim externý náhľad bez akejkoľvek hardvérovej akcelerácie, tiež ako prvý umožnil editovanie so snímkovaním 24p. Má natívnu podporu pre sieťový rendering. Dokáže rozdeliť renderovaný súbor na malé časti, ktoré rozdistribuuje medzi všetky renderovacie uzly (renderovacie počítače), ktoré ich renderujú separátne a hotové časti sa posielajú späť, z hotových vyrenderovaných časti sa poskladá výsledná video sekvencia.

Vlastnosti:

- neobmedzený počet stôp, obmedzujúci faktor je výkon počítača
- podpora rôznych pomerov strán (4:3, 16:9, …)
- zmena parametrov sekvencie v hociktorom okamžiku editácie
- silné nástroje pre korekciu farieb
- podpora externého náhľadu cez FireWire, LCD, DVI
- najširšia podpora kodekov
- spracovanie zvuku v kvalite až 24-bit/192KHz

3.2 Voľne dostupné nástroje

3.2.1 LiVES

LiVES vznikol v roku 2002 ako video editačný systém pre Linux. Od vtedy už funguje na viacerých operačných systémoch. Jeho cieľom je byť jednoduchý na používanie a zároveň byť mocným nástrojom.

Jeho nevýhoda je podobná ako pri väčšine opensource video editoroch, a to že sa nehodí na profesionálnu prácu s videom. Momentálne je jeho vývoj pozastavený pre nedostatok finančných prostriedkov.

Vlastnosti:

- podpora viacerých stôp
- možnosť zachytenia obsahu akéhokoľvek okna, platí pre X11
- podpora JACK prenosov medzi aplikáciami
- podpora efektov
- zabudované skriptovacie jadro

3.2.2 FFmpeg

FFmpeg je veľmi rýchly audio a video konvertor. Je postavený a zároveň aj vyvíjaný ako súčasť veľmi výkonnej knižnice kodekov FFmpeg, ktorej vlastnosti patrične využíva. Ovládanie tejto utility je riešené prostredníctvom príkazového riadku, pretože nemá grafické rozhranie (GUI), to však nie je jeho nevýhodou, ale naopak výhodou (automatizácia úloh, jednoduché skriptovanie, ...). Jeho vývoj neustále pokračuje, za čo vďačí neustálemu vývoju knižnice FFmpeg.

Vlastnosti:

- konverzia videa z jedného kodeku do druhého
- zmena vzorkovacej frekvencie audia
- zmena rozlíšenia videa
- extrakcia jednotlivých stôp (audio/video stream-ov)
- vytvorenie videa zo sekvencie obrázkov
- pridanie/odstránenie audia k videu a naopak
- pridanie titulkov do filmu
- široká podpora kodekov vďaka knižnici FFmpeg
- zachytávanie videa z pracovnej plochy (X11)
- zachytávanie videa zo zariadení

3.2.3 VirtualDub

VirtualDub je výkonný nástroj na nahrávanie a spracovávanie videa pod operačným systémom Windows. Je šírený pod licenciou GPL, takže jeho zdrojové kódy sú voľne dostupné. Program postráda editačné možnosti veľkých strihacích štúdií ako je napr. Adobe Premiere, zato je orientovaný na rýchle lineárne operácie s videom. Je rozšíriteľný o externé video filtre. Dovoľuje dávkové spracovanie videa cez vstavaný skriptovací jazyk, čo je vhodné pri veľkom počte spracovávaných súborov. Medzi podporované formáty patria AVI, MPEG (čítanie) a sada BMP obrázkov.

Zaujímavou možnosťou VirtualDub-u je frameserver, ktorý umožňuje predať výstup VirtualDub-u ďalšiemu programu bez nutnosti vytvárania výstupného súboru. Je to užitočné napr. ak použitý program nepodporuje daný formát (napr. AVI) a VirtualDub ho podporuje, vtedy je možné tento súbor načítať vo frameserver-i a jeho výstup poslať do tohto programu.

4 Štruktúra video súborov

Oblasť zaoberajúca sa archivovaním video sekvencií je veľmi obsiahla, preto táto kapitola obsahuje len jemný úvod do tejto problematiky. Vysvetľuje základné pojmy patriace do tejto oblasti, ako sú používané farebné priestory, čo je to kodek a čo kontajner.

4.1 Kodek, kódovanie a dekódovanie

V súčastnosti so zvýšením výkonu osobných počítačov a s rozširovaním sa internetu je používanie kodekov každodennou záležitosťou, po väčšinou sa s nimi stretávame pri pozeraní filmov (divx, xvid, quicktime, wmv), počúvaní hudby (mp3, ogg), či prezeraní obrázkov (jpeg, png) na internete. Kodek (anglicky codec – coder/decoder, compressor/decompressor) je program schopný kódovania a dekódovania vstupného digitálneho dátového signálu (prúdu bytov). Kodek môže pracovať ako kodér, kedy dátový tok zakóduje do vhodného formátu pre prenos, uloženie alebo kompresiu, alebo ako dekodér, kedy dátový tok dekóduje na prehliadanie alebo editáciu. Toto kódovanie sa vykonáva z viacerých dôvodov, väčšinovým motívom je úspora miesta na disku vzhľadom na to že nezakódované video ma extrémne nároky na diskový priestor. Doteraz sa vymysleli stovky kodekov počínajúc kodekmi zadarmo, končiac kodekmi s cenou stoviek dolárov.

Kodeky rozdeľujeme do dvoch skupín:

- stratové (lossy, napr. mp3, divx)
- bezstratové (lossless, napr. ogg, msu)

Stratové kodeky dosahujú lepší kompresný pomer oproti bezstratovým, ale cennou za lepší pomer kompresie je zhoršenie kvality audia/videa. Našťastie tieto kodeky orezávajú informáciu (kvalitu) na miestach kde si túto stratu všimneme nepatrne, pričom využívajú špecifické vlastnosti ľudského zraku, sluchu a kódovaného audia/videa. Napr. pre záznam futbalového zápasu je dôležité neskresľovať pohyb, ktorý sa rýchlo mení, ale na druhú stranu 100% vernosť farieb nehrá dôležitú rolu. K stratovým kodekom sa viaže jeden problém a to ten, že sú postavené na modele vnímania informácie človekom, a je veľmi obtiažne a pravdepodobne i nemožné objektívne zmerať mieru straty informácie. Klasické metódy vyjadrenia chyby v tomto prípade dávajú nezmyselné výsledky. Napríklad pri porovnaní priebehu originálneho zvukového záznamu a záznamu MP3, môže nastať situácia, že ani jedna hodnota sebe si odpovedajúcich samplov nezostane zachovaná, ale priemerný človek si pri počúvaní nevšimne rozdiel medzi originálnou nahrávkou a komprimovanou kópiou. V záujme dosiahnuť čo najmenší objem výsledných dát je väčšina kodekov stratová.

Bezstratové kodeky využívame ak sa informačná hodnota audia/videa nesmie stratiť, požadujeme najvyššiu možnú kvalitu a zároveň chceme ušetriť pár percent diskového priestoru. Dalo by sa povedať že vždy dosahujú horších kompresných pomerov oproti stratovým kodekom.

Na nasledujúcich obrázkoch je zobrazený efekt stratovej kompresie. Ľavý obrázok (1) je originál bez kompresie a naň sa aplikuje JPEG kompresia so zachovaním 7% kvality originálu (7% je zvolených pre zvýraznenie degradácie obrazu, väčšinou sa používa kvalita okolo 90%, pri takejto kvalite si ľudské oko nevšimne skoro žiadny rozdiel medzi výsledným a pôvodným obrazom), výsledok vidíme na pravom obrázku (2).

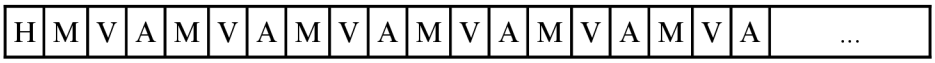
Obrázok 1: bez kompresie, veľkosť 29346 bytov

Obrázok 2: JPEG kompresia, kvalita 7%, veľkosť 1056 bytov

4.2 Kontajner

Väčšina video sekvencií obsahuje audio a video súčasne plus určité metadáta pre synchronizáciu audio dát s videom. Každý z týchto dátových tokov je obsluhovaný odlišným kodekom (video divx, audio mp3), ale pre spracovanie či prehrávanie je potrebné aby tieto toky (streamy) boli uložené spolu v prenositeľnej forme, v tzv. kontajnery. Jednotlivé dáta audio a video tokov sa v kontajnery priebežne striedajú, a vytvárajú pakety, každý paket má v sebe minimálne identifikátor ku ktorej stope (streamu) patrí. Častou chybou je označovanie formátu AVI ako kodeku, AVI nie je kodek, ale kontajner, ktorý môžu využívať rôzne kodeky. Medzi ďalšie kontajnerové formáty patria napr. ASF, MOV, MPG, RM.

Typický obsah video sekvencie je zobrazený na nasledujúcich diagramoch:



Vysvetlivky:

H – hlavička súboru

M – informačné metadáta

V – video dáta (video paket)

A – audio dáta (audio paket)

Ako príklad kontajnerového formátu popíšem jeden z najrozšírenejších kontajnerov AVI – Audio Video Interleave, vyvinutý firmou Microsoft v roku 1992 ako časť technológie Video pre Windows. AVI súbory môžu obsahovať audio aj video dáta a umožňujú synchrónne prehrávanie audia s videom. Súbor môže obsahovať viacej audio a video streamov, avšak táto vlastnosť sa využíva sporadicky. AVI je špeciálny prípad RIFF súboru, ktorý rozdeľuje dáta do malých blokov (chunks). Každý blok je označený 32 bitovým identifikátorom (FOURCC - four character code). Štruktúru súboru tvorí jeden hlavný blok (značka 'AVI '), tento blok sa skladá z dvoch povinných blokov a jedného voliteľného bloku. Prvý povinný blok (značka 'hdrl') obsahuje informačnú hlavičku (rozlíšenie videa, dátový tok, …). Druhý blok označený ako 'movi' obsahuje audio/video dáta. Voliteľný blok (značka 'idx1') slúži na indexáciu dátových blokov vo vnútri súboru. AVI formát nieje viazaný na konkrétne kodeky, pre audio a video v AVI súbore môže byť kódované virtuálne akýmkoľvek kodekom.

V porovnaní s modernými kontajnerovými formátmi (napr. Matroska) AVI zaostáva, napriek tomu predstavuje jeden z najpopulárnejších formátov súčasnosti, pravdepodobne kvôli vysokej kompatibilite s video prehrávačmi. V júni 2005, DivX inc. uvoľnil nový kontajnerový formát nazvaný DivX Media Format (koncovka .divx) ako nástupcu dvojičky AVI kontajner + DivX kodek. Napriek faktu, že tento formát je vlastne rozšírený AVI formát (založený na podobnej RIFF štruktúre), nezaznamenal patrné rozšírenie.

4.3 Farebné priestory RGB a YC_bC_r

Najznámejším formátom v počítačoch je RGB skladajúci sa z troch zložiek (kanálov) červenej, zelenej a modrej. Na tomto princípe funguje napr. monitor, televízia. Každý bod, ktorý vidíte sa skladá práve z červeného, zeleného a modrého luminoforu, ktoré sú blízko seba. Pretože sú veľmi blízko, oko ich vníma ako jeden farebný bod. Televízia tiež zobrazuje video pomocou RGB luminofórov, ako je popísané vyššie, ale televízny signál nie je prenášaný v RGB formáte. V dobe

keď vznikla televízia pracovala čiernobielo, čo znamená že pre určenie jednotlivého bodu na obrazovke je potrebná len jedna informácia.

Keď začala byť vyvíjaná farebná televízia, bol vznesený požiadavok na to aby farebný signál mohli sledovať aj majitelia čiernobielych televízií, pretože milióny divákov svoju čiernobielu televíziu nevyhodia. Namiesto toho aby sa signálom šíril obraz vo formáte RGB, bola použitá technológia nazývaná ako YCC. Y je rovnaká zložka ako jediná zložka v čiernobielom signále, dve C sú nové farebné zložky. Dve farebné zložky určujú farbu bodu a Y jeho jas. Týmto bola zaistená kompatibilita signálu s čiernobielymi televíziami.

Model YCC sa nepoužíva len v televíznom signále, ale aj v digitálnom videu, pretože jeho vlastnosti a vlastnosti ľudského oka umožňujú efektívnu kompresiu videa. Využíva sa poznatok, že ľudské oko je citlivejšie na zmenu jasu než na zmenu farby, takže namiesto toho aby sa ukladali všetky informácie zložiek YCC, ukladá sa len časť. Pre profesionálnu kvalitu stačí uloženie len polovice farebných zložiek narozdiel od zložky Y, táto kompresia sa nazýva 4:2:2. Často používané je kódovanie 4:2:0, ktoré znižuje farebné rozlíšenie vo vertikálnom aj horizontálnom smere. Farebná informácia je štvrtinová oproti pôvodnej. Zníženie farebného rozlíšenia môže mať v určitých prípadoch za následok viditeľné artefakty.

5 Knižnice na prácu s videom

Multimédiá hrajú vo svete počítačov a operačných systémov jednu z kľúčových rolí. Žiadny výrobca operačného systému pre bežných užívateľov si nedovolí vynechať ich podporu, a ani jedna z najrozšírenejších platforiem (Window, Unix/Linux, MacOS) nezabúda na podporu digitálneho videa, či už je to Video pre Windows na Windows, FFmpeg na Unix-e alebo QuickTime na MacOS.

5.1 FFmpeg

FFmpeg predstavuje kompletné opensource riešenie pre nahrávanie, konvertovanie a streaming audia a videa. Je vyvíjaný na platforme Linux-e, tiež si skvele rozumie s väčšinou Unix-ov (FreeBSD, NetBSD, BeOS), ba čo viac pracuje aj na operačnom systéme MS Windows.

FFmpeg pozostáva z dvoch hlavných a pár pomocných častí. Prvou hlavnou časťou je knižnica **libavcodec** predstavujúca zbierku audio a video kodekov, nájdeme v nej všetky v súčastnosti významnejšie kodeky (H.264, XviD, MPEG4, MP3, ...). Zahrňuje v sebe aj proprietárne kodeky (Windows Media Audio/Video, ...), ku ktorým neexistuje voľne prístupná technická špecifikácia. Pretože k nim neexistuje voľne prístupná dokumentácia, jej vývojári musia s pomocou reverzného inžinierstva pracovať na zistení princípov daných kodekov. Kvôli tomuto problému nie je podpora niektorých proprietárnych kodekov sto percentná. Zbierka kodekov je vskutku obrovská, FFmpeg obsahuje okolo 170 kodekov a rozhodne sa nezastaví pri tomto čísle.

Jednu časť knižnice tvoria kodeky zastúpené knižnicou **libavcodec**, druhú predstavuje knižnica na zaobchádzanie s kontajnerovými formátmi **libavformat**. Ako už bolo spomenuté, multimediálne súbory obsahujú jednak audio stopy a jednak video stopy, tieto dáta sú uložené v kontajnery (súbore s určitým formátom) a sú rozdelené do paketov. Preto ak pracujeme s multimediálnym súborom pracujeme vlastne s kontajnerom obsahujúcim pakety audio/video dát (film obsahuje video a audio stopu, ktorých dáta sa priebežne pravidelne striedajú aby umožnili plynulé prehrávanie filmu cez jeden prechod súborom). Na prácu (čítanie/zápis) s kontajnermi je určená knižnica **libavformat**. Táto knižnica pokrýva vyše 100 kontajnerových formátov vrátane formátov AVI, Matroska, MOV, ASF.

K pomocným častiam patria **libavutil** - zbierka pomocných rutín spoločných pre viaceré časti knižnice, **libpostproc** – knižnica pre postprocessing videa a **libswscale** – rutiny na zmenu veľkosti obrázku.

Práca s FFmpeg knižnicou je jednoduchou záležitosťou, ale začiatočníkom veľmi sťažuje prácu nedostatok kvalitnej dokumentácie (takmer žiadna) a samotný vývojári vás väčšinou odkážu priamo na zdrojové kódy knižnice (no nie celá knižnica je okomentovaná). Zjednodušuje vývojárom prácu tým, že všetky kodeky a kontajnery obaľuje jednotným API. Využíva výhody objektovo

orientovaného programovania i keď je napísaná v programovacom jazyku (je písaná v jazyku C), ktorý nemá priamu podporu objektov.

Na tejto knižnici je založených pár vynikajúcich programov, napr. MPlayer (jeden z najlepších video prehrávačov) alebo VLC. Svoju budúcnosť má ešte len pred sebou, jej vývoj pokračuje, pracuje sa na zdokonalení stávajúcich kodekov, ako aj pridávaní nových kodekov. Svedčí o tom aj fakt, že FFmpeg sa aktívne zapája do programu Google Summer Of Code.

5.2 Video pre Windows

Video pre Windows je multimediálny framework vyvinutý firmou Microsoft, ktorý umožnil operačnému systému Microsoft Windows prehrávať digitálne video. Svetlo sveta uzrel v roku 1992, ako reakcia na technológiu QuickTime, ktorá umožnila prehrávanie digitálneho videa na počítačoch Macintosh. S počiatku existoval ako prídavný balíček do Windows 3.1, neskôr s rastúcim významom digitálneho videa sa stal integrálnou súčasťou všetkých operačných systémov Microsoft Windows.

Je možné ho rozdeliť na tri veľké časti:

- súborový formát určený pre digitálne video (AVI)
- programátorské API
- nástroje na prehrávanie, nahrávanie a editáciu

Pôvodná implementácia mala množstvo nedostatkov zahrňujúcich napr. maximálne rozlíšenie 320x240 a maximálna snímková frekvencia 30 obrázkov za sekundu. Dnes už týmito nedostatkami netrpí.

6 Strihacie jadro

Táto kapitola je venovaná hlavnej časti tejto bakalárskej práce, ktorou je návrh a implementácia jadra na strih video sekvencií. Podrobne je vysvetlená architektúra a princípy fungovania tohto strihacieho jadra, opísané sú tu všetky jeho komponenty. Spoznanie návrhu je nevyhnutnosťou pre jeho neskoršie používanie, či sa jedná o jeho využitie ako knižnice v iných aplikáciách, jeho rozšírenie (pluginy), alebo úpravu jeho zdrojových kódov. Časť implementácia pojednáva o nástrojoch použitých na vytvorenie tohto softvéru. Hlavnou témou tejto podkapitoly je vývojové prostredie GNUstep, ktoré poskytuje základné stavebné kamene pre tvorbu aplikácii (podobná funkcionalita sa dá získať napr. použitím štandardnej knižnice šablón v C++). Neoddeliteľnou vlastnosťou tohto jadra je veľká rozšíriteľnosť, samotné jadro dokonca vyžaduje určité rozširujúce moduly (napr. neobsahuje v sebe žiadny kodek, tie sú do neho pridávané vo forme pluginov, tiež neobsahuje žiaden audio/video filter). Súčasťou tejto práce nie je iba vytvorenie samotného jadra, ale aj ukážkových aplikácií na demonštráciu jeho funkčnosti, o tejto téme pojednávajú zvyšné podkapitoly.

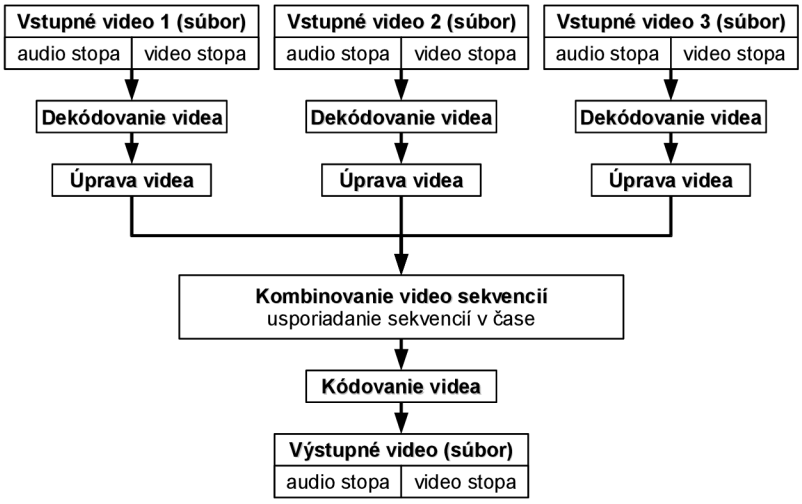
6.1 Návrh a architektúra

Návrh architektúry rešpektuje jednak požiadavky kladené na schopnosti strihacieho jadra, tak tiež a zohľadňuje možnosti použitých vývojových prostriedkov.

Na nasledujúcom diagrame (3) je naznačený približný postup spracovania jednej video sekvencie. Ako vstup je použité jedno video v ľubovoľnom formáte (pre dosiahnutie čo najvyššej výstupnej kvality a rýchlosti spracovania sa odporúča vstupný formát bez kompresie, príp. s bezstratovou kompresiou a čo najvyšším možným rozlíšením audia a videa). Vstupné video sa dekóduje, čiže prevádza z toku bytov na obrázky a sample, ktoré sa ďalej ľahko spracovávajú. V ďalšom kroku sa na tieto dáta aplikujú rôzne úpravy ako je napr. orezanie stopy, pretiahnutie stopy, echo efekt, … Po požadovaných úpravách sa výsledne video teraz ešte vo forme obrázkov a samplov znovu kóduje do toku bytov (audio a video stôp, výslednú kvalitu kompresie volíme podľa účelu použitia, napr. pre streaming zvolíme nižšiu kvalitu ako pre film na DVD). Výsledné upravené a kódované video je uložené v súbore. Tento diagram pre zjednodušenie znázorňuje úpravu len jedného videa. Prax vyžaduje pokročilejšiu úpravu videa, ktorá sa odlišuje od tejto

Obrázok 3: Jednoduché spracovanie videa

jednoduchej napr. tým že zahrňuje viacero vstupných súborov. Pokročilé spracovanie videa je obsahom obrázku číslo 4.



Obrázok 4: Pokročilé spracovanie videa

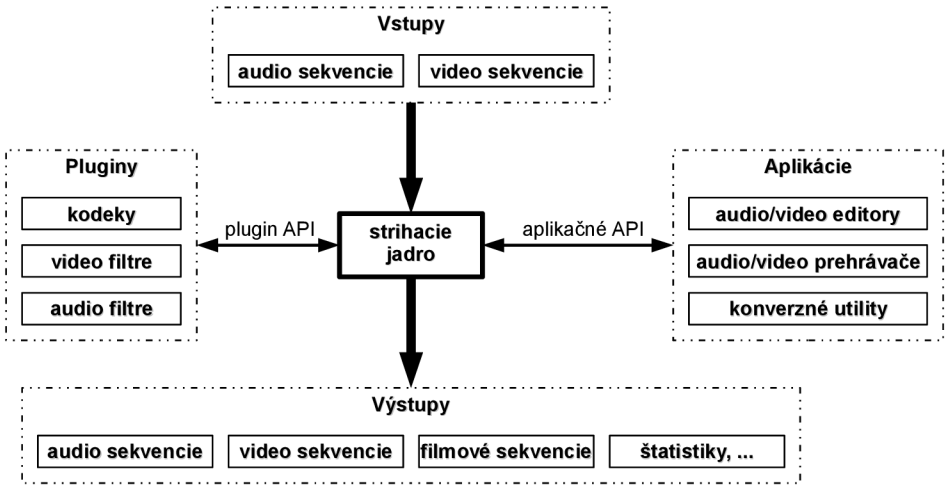
Zložitejšie spracovanie videa v podstate vychádza z jednoduchého spracovania, hlavným vylepšením je spracovanie viacerých vstupných sekvencií. Počiatočné fázy sú zhodné (dekódovanie, úprava) s jednoduchou úpravou, zmena je v momente kedy by sa malo výsledné video zakódovať. Pretože sa pracuje s viacerými vstupnými a jedným výstupným súborom, je nevyhnutnosťou usporiadanie vstupných upravených sekvencií na časovej osi. Usporiadané sekvencie sú potom správne umiestnené vo výslednej výstupnej sekvencii, a táto je následne zakódovaná do výstupného súboru. Presne na takomto istom modeli spracovania video sekvencií je postavené strihacie jadro, ktorému sa venuje táto práca. Tento model je síce zložitejší, ale aj výkonnejší a tiež je ďalej rozšíriteľný napr. o filtre pracujúce s viacerými vstupnými sekvenciami.

Po načrtnutí modelu strihu videa venujme pozornosť návrhu strihacieho jadra. V súčastnosti je modernou a praxou overenou metódou návrhu objektovo orientované programovanie (OOP). OOP počíta s rozdelením programu na kolekciu vzájomne komunikujúcich objektov (rozdelenie na komunikujúce objekty je inšpirované reálnym svetom, ktorý sa skladá z objektov komunikujúcich jeden s druhým). Použiť OOP je možné aj v ne objektovo orientovaných programovacích jazykoch (dobrým príkladom je knižnica FFmpeg, písaná v ne objektovo orientovanom jazyku C, avšak využívajúca výhody OOP), ale pre získanie všetkých vymožeností OOP a urýchlenie vývoja, by sme si mali vybrať jeden objektovo orientovaný jazyk, napr. Objective-C, C++ alebo Java.

Jedným z kritických faktorov strihacieho jadra je jeho výkon, predsa by sme nechceli aby výpočet jedného obrázku trval 1 hodinu či viac (pričom v konkurenčnom programe trvá obdobný výpočet 1s). Jeho výkon závisí predovšetkým od kvality jeho návrhu a použitých algoritmov. Čiastočne jeho výkonnosť ovplyvňuje použitý programovací jazyk (softvér napísaný v basic-u je pomalší ako ekvivalent napísaný v jazyku C), ale predovšetkým je to návrh, pri ktorom treba na tento

faktor klásť veľký dôraz a počítať s ním od počiatku. Vhodné sú predovšetkým kompilované jazyky ako sú napr. C, C++, Objective-C.

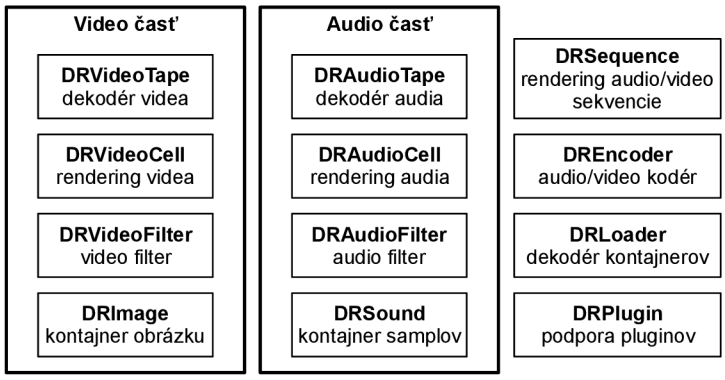
Dobre navrhnuté strihacie jadro je minimalistické s čo najmenším počtom závislostí (napr. nezávislosť použitého kodeku). Tieto vlastnosti sú ovplyvnené výberom framework-u (v tomto prípade GNUstep) na ktorom bude postavené a umožnením rozšírení prostredníctvom pluginov (podpora kodekov, audio/video filtrov). Táto architektúra je vyobrazená na obrázku č.5.



Obrázok 5: Architektúra strihacieho jadra

Medzi vstupy patria multimediálne súbory v rôznych formátoch, ich načítanie (dekódovanie) majú na starosti externé dekodéry (napr. FFmpeg plugin obsahuje jeden, opísaný o pár sekcií ďalej). Komunikácia s pluginmi je zabezpečená pomocou plugin API, a pretože je toto jadro nezávislé na použitom kodeku a zároveň žiadny kodek neobsahuje je nutné k nemu požadovaný kodek pripojiť ako zásuvný modul. Výstupom je sekvencia, ktorá môže ale nemusí obsahovať video aj audio stopu, môže obsahovať len jednu z nich, alebo žiadnu ak výstupom sú štatistiky. Celý proces strihu je riadený prostredníctvom aplikačného API, cez ktoré aplikácie (napr. video editor) komunikujú s jadrom.

V nasledujúcom texte tejto podkapitoly sa budeme venovať jednotlivým komponentom strihacieho jadra zodpovedným za spracovania audia, videa, sekvencií, podporu pluginov, … Stručný prehľad aj s krátkym popisom ich zamerania nájdete na nasledujúcom obrázku.

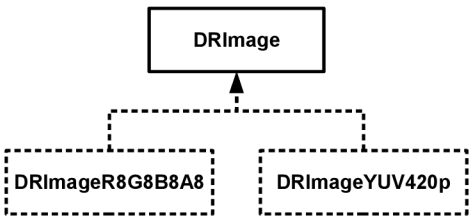


Obrázok 6: Komponenty strihacieho jadra

6.1.1 DRImage

Pri spracovaní video sekvencií sa neustále pracuje s obrázkami a zvukovými samplami, ktoré sa presúvajú (po spracovaní) z jednej komponenty do druhej, až k cieľovej komponente, ktorou je enkodér. Robustnosť týchto prvkov ovplyvňuje kvalitu celej architektúry, pretože patria do najnižšej úrovne návrhu (preto im musí byť venovaná špeciálna pozornosť).

V svete počítačovej grafiky prevláda grafický formát RGB skladajúci sa z troch farebných zložiek a to z červenej (**r**ed), zelenej (**g**reen) a modrej (**b**lue) s ôsmymi bitmi na každú zložku. Úpravy obrázku sú v tomto formáte jednoduché, intuitívne a rýchle, čo sa môže na prvý pohľad zdať ako vynikajúci základný formát. Jednoduché rozšírenie RGB obrázku je pridanie priehľadnosti každému pixlu, tzv. alfa kanál. RGB s alfa kanálom sa nazýva RGBA. Teraz by sa nám mohlo znovu zdať, že všetky vlastnosti na základný formát máme vyriešené a cieľový formát bude RGBA. Pri detailnejšom pohľade na kódovanie videa zistíme, že väčšina kodekov vyžaduje formáty Ycc (napr. YUV420p). Požiadavka dvoch formátov RGBA (vhodný pre spracovanie a úpravu) a Ycc (podporovaný kodekmi) časom narastie na požiadavku troch a viacerých formátov (RGBA s 32bitmi na zložku, …). Problém viacerých formátov elegantne vyriešime použitím abstraktnej triedy **DRImage** (obrázok č.7), od ktorej odvodíme žiadané formáty. Implementácia odvodených tried ej súčasťou strihacieho jadra. Rozšírenie jadra o nový formát počíta s úpravou zdrojových kódov jadra (pridanie identifikátoru formátu, triedy formátu a informačnej štruktúry). Štandardnou súčasťou je formát RGBA reprezentovaný triedou DRImageR8G8B8A8. Táto trieda je implementáciou formátu RGBA s ôsmymi bitmi na každý kanál. Pretože je toto zatiaľ jediný podporovaný formát, musia všetky video komponenty komunikovať výhradne prostredníctvom tohto formátu.

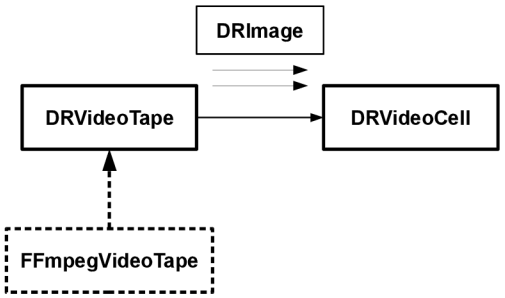


Obrázok 7: Trieda DRImage

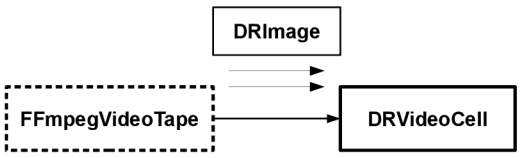
Na popis formátu obrázku slúži štruktúra DRImageFormat, obsahujúca identifikátor typu obrázku a informácie špecifické pre každý formát zvlášť (pre R8G8B8A8 je to položka r8g8b8a8).

6.1.2 DRVideoTape

DRVideoTape reprezentuje kontajner obrázkov zo sekvenčným prístupom. V reálnom svete je to obdoba video klasickej pásky. Predstavuje komunikačné rozhranie dekodéra, dekodéra video súboru na sekvenciu obrázkov. Pretože je to iba rozhranie je implementovaný ako abstraktná trieda (v C++ trieda s aspoň jednou čiste virtuálnou metódou, v Objective-C máme dve možnosti, prvá je použitie triedy s prázdnou implementáciou metód, druhá je nahradenie triedy protokolom). Súčasťou implementácie jadra nie je ani jedna konkrétna podtrieda DRVideoTape (jadro je formátovo nezávislé).



Obrázok 9: Komunikácia triedy DRVideoTape v systéme



Obrázok 8: Komunikácia podtriedy FFmpegVideoTape v systéme

DRVideoTape definuje metódy na ovládanie video pásky a prístup k jednotlivým snímkom. Dovoľuje použiť kodeky s pevnou snímkovou frekvenciou, ale aj s premenlivou. V celkovom systéme komunikuje iba s triedou DRVideoCell (popísanou nižšie). Pri vytváraní konkrétnej podtriedy platia isté zásady a to:

- podtrieda musí implementovať všetky metódy triedy DRVideoTape
- pri čítaní obrázka (readImage) musí vracať korektné hodnoty timestamp a duration, musí platiť že $timestamp_i + duration_i = timestamp_{i+1}$
- metóda endOfTape vracia YES až po prečítaní posledného obrázka z pásky
- inicializácia triedy DRVideoTape je cez metódu init
- celá páska má jednotný formát obrázkov

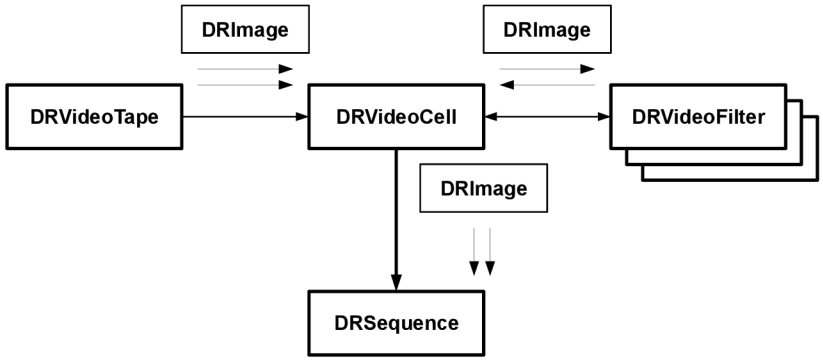
S triedou DRVideoTape je spätá štruktúra DRVideoFormat slúžiaca na popis formátu pásky (formát obrázkov, dĺžka video pásky, …), jej detailnejší popis je v súbore DRVideoTape.h.

6.1.3 DRVideoFilter

V procese spracovania videa sú na video sekvencie aplikované rôzne video efekty, prostredníctvom filtrov. Podporou filtrov v tomto strihacom jadre je trieda DRVideoFilter. Je to abstraktná trieda, ktorá definuje komunikačné rozhranie video filtru. Každý video filter musí byť odvodený z tejto triedy, a musí implementovať jej rozhranie. Definícia je umiestnená v súbore DRVideoFilter.h. Jej zaradenie v celkovom systéme je patrné z obrázku č.10.

6.1.4 DRVideoCell

DRVideoCell alebo video renderovacia bunka, je jednou z kľúčových komponentov celého systému. Implementuje v sebe logiku ovládania video pásky, logiku riadenia video filtrov a svoju vlastnú logiku. DRVideoCell nieje samostatná trieda, ale je prepojená s ostatnými triedami (DRVideoTape, DRVideoFilter), priebeh komunikácie s okolitými prvkami je vyobrazený na nasledujúcom obrázku.



Obrázok 10: Komunikácia video renderovacej bunky

Z neho je patrný jej vzťah k ostatným prvkom v systéme. Jednosmerné šípky smerom od video pásky značia tok (čítanie) obrázkov z video pásky smerom do bunky. Obrázok načítaný z video pásky je prenechaný prideleným video filtrom na spracovanie (aplikovanie efektov). Video filtre upravia zdrojový obrázok a pošlú ho naspäť renderovacej bunke, toto je už výsledný obraz, ktorý ďalej putuje na spracovanie do sekvencie (DRSequence). Implementácia DRVideoCell podporuje neobmedzený počet video filtrov.

Korektné vytvorenie a použitie renderovacej bunky popisuje nasledujúci úsek kódu:

```
// nacitanie videa
FFmpegVideoTape *tape = [FfmpegVideoTape
                         videoTapeWithFile:@"video.avi"
                         streamIndex:0];
// vytvorenie bunky
DRVideoCell *cell = [DRVideoCell cellWithTape:tape];
```

```
// uvolnenie pasky

// najdenie a vytvorenie instancie filtra
Class filterClass = DRFindVideoFilterClass(@"MotionBlur");
DRVideoFilter *filter = [filterClass new];

// pripojenie filtra k bunke
[cell addVideoFilter:filter];
RELEASE(filter);
// nasleduje nacitanie jedneho obrazku
DRRenderContext rc;
rc.cellPosition = 5000000; // 5 sekund
rc.sequencePosition = 5000000; // 5 sekund

// nacitame obrazok
DRImage *image = [cell renderWithContext:&rc];
```

6.1.5 DRSound

Takisto ako je trieda DRImage základnou triedou na prenos informácií vo video subsystéme strihacieho jadra, tak podobnú funkciu v zvukovom subsystéme hrá trieda DRSound. Na rozdiel od DRImage, ktorá vždy obsahovala jeden obrázok a bola abstraktnou triedou, je DRSound kontajner samplov a nieje abstraktnou triedou, nie je ani určená na tvorenie hierarchie dedičnosti.

Podporuje rôzne formáty audia (rôzny počet kanálov, rôzna frekvencia, ...), ktoré sa zadávajú parametrom pri jej vytváraní.

6.1.6 DRAudioTape

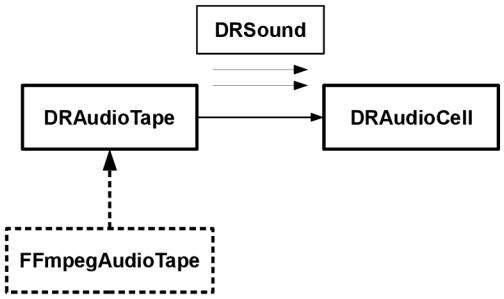
DRAudioTape je kontajner samplov zo sekvenčným prístupom. Objekt s podobnými vlastnosťami je skutočná audio páska. Tvorí komunikačné rozhranie dekodéra audia, dekodéra audio súboru na sekvenciu samplov. Implementačne je to abstraktná trieda predurčená na dedenie. Súčasťou implementácie jadra nie je ani jedna konkrétna podtrieda DRAudioTape (podobne ako DRVideoTape).

DRAudioTape popisuje metódy ovládajúce audio pásku a prístup k jednotlivým samplom. V celkovom systéme komunikuje iba s triedou DRAudioCell (popísanou nižšie), ale je použiteľná aj samostatne. Pri vytváraní konkrétnej podtriedy platia isté zásady a to:

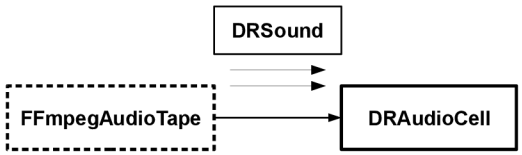
- podtrieda musí implementovať všetky metódy triedy DRAudioTape
- pri čítaní obrázka (readSamples) musí vracať korektné hodnoty start a end, musí platiť

$end_i = start_{i+1}$

- metóda endOfTape vracia YES až po prečítaní posledného samplu
- inicializácia triedy DRAudioTape je cez metódu init
- celá páska má jednotný formát samplov



Obrázok 11: Komunikácia triedy DRAudioTape v systéme



Obrázok 12: Komunikácia podtriedy FFmpegAudioTape v systéme

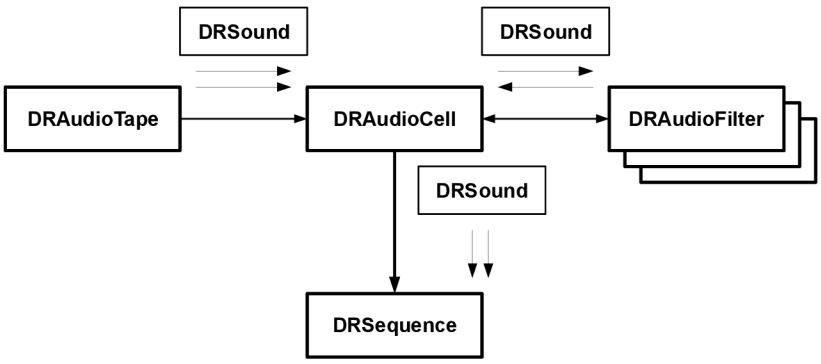
S triedou DRAudioTape je spätá štruktúra DRSoundFormat slúžiaca na popis formátu pásky.

6.1.7 DRAudioFilter

Podobne ako video má filtre, aj na audio je možné aplikovať filtre. Na ich podporu slúži táto trieda. Jej význam a práca s ňou je podobná triede DRVideoFilter. Definícia s detailnejším popisom sa nachádza v súbore DRAudioFilter.h. Jej zaradenie v celkovom systéme je patrné z obrázku č.13.

6.1.8 DRAudioCell

DRAudioCell je obdoba video renderovacej bunky, ale na spracovanie audia. Implementuje podobnú logiku ako video bunka. Je je závislá od tried DRAudioTape (kontajner samplov) a DRAudioFilter. Priebeh komunikácie s okolitými prvkami je vyobrazený na nasledujúcom obrázku.



Obrázok 13: Komunikácia audio renderovacej bunky

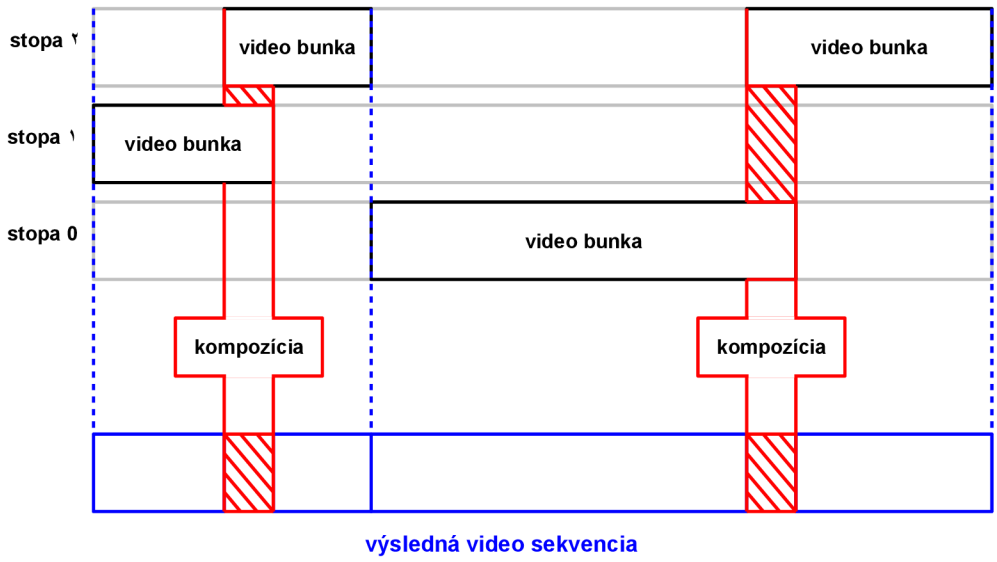
Priebeh komunikácie je nasledovný, najprv sa načítajú požadované sample z audio pásky, na ne sa aplikujú filtre a takto upravené sample sa predajú sekvencii na umiestnenie do výsledného súboru.

6.1.9 DRSequence

Predošlé podkapitoly opisovali základné stavebné bloky sekvencie ako sú audio/video pásky, audio/video filtre a audio/video renderovacie bunky produkujúce sekvencie samplov/obrázkov, ktoré ešte musia byť umiestnené na správne miesto do výslednej sekvencie. DRSequence je trieda poskytujúca objekty s takouto funkcionalitou.

Výsledná video sekvencia obsahuj jednu audio a jednu video stopu, ale pre strih je výhodnejšie pracovať s viacerými audio a video stopami, ktoré sa pri spracovaní kombinujú. Vďaka kombinovaniu stôp sa dajú do výslednej sekvencie vkladať rôzne efekty. Na nasledujúcom obrázku je znázornený koncept viacerých pracovných video stôp a jednej výslednej video stopy, obdobne to platí aj pre audio stopy.

Pri vytváraní sekvencie sa nastaví jej výstupný formát, tento formát musia mať všetky bunky, ktoré do nej vkladáme, v opačnom prípade sa strih nepodarí.



Obrázok 14: Princíp video stôp

Spôsob kombinovania jednotlivých stôp závisí od použitého formátu obrázkov, napr. obrázky RGB neumožňujú žiadne kombinovanie, proste obrázok v najvyššej stope je zapísaný do výslednej sekvencie. Naproti tomu RGBA dovoľuje rôzne prechody medzi stopami (formát RGBA je súčasťou jadra), čo sa využíva pri video filtroch.

6.1.10 DRPlugin

DRPlugin je abstraktná trieda (implementovaná ako protokol) určená na vytváranie rozšírení. Každý rozširujúci modul musí mať triedu s takýmto rozhraním. Obsahuje dve hlavné metódy a to jednu ktorá sa volá pri načítaní pluginu (registerPlugin) a druhá je volaná pri odregistrovaní pluginu (unregisterPlugin). Metóda registerPlugin je volaná pri nahrávaní rozšírenia (DRFoundationInit) a jej účelom je registrácia všetkých komponent (enkodéry, kodéry, filtre, …) pluginu do systému. Opakom metódy registerPlugin je unregisterPlugin, ktorá je volaná pri odstraňovaní rozšírenia zo systému (najčastejšie pri volaní DRFoundationQuit). Vytváranie pluginov je popísané v sekcii Plugin.

6.2 Implementácia

V tejto časti je popísaná implementácia návrhu strihacieho jadra, ktorému sa venovala predchádzajúca kapitola. Sú tu opísané použité vývojové prostriedky a to predovšetkým GNUstep framework, na ktorom je postavené celé jadro.

Návrh architektúry vychádza z objektovo orientovaného programovania, čiže je objektový. Z tohto dôvodu je implementácia vykonaná v programovacom jazyku s podporou objektov. Ako vhodné jazyky sa javia C++ a Objective-C. Ich nespornou výhodou oproti ostatným jazykom (C# a Java) je že patria do kategórie kompilovaných jazykov (teda ponúkajú vyšší výkon). Pre oba spomínané jazyky existuje kvalitná tzv. kolekcia základných tried (pole, slovník, …), či Standard Template Library v C++ alebo Foundation v Objective-C. Na implementáciu strihacieho jadra bol zvolený jazyk Objective-C z viacerých dôvodov (samozrejme je možné ho implementovať aj v iných jazykoch):

- čistejšia objektová orientácia
- natívna podpora pluginov v Foundation framework-u
- transparentná podpora distribuovaných objektov (možnosť využitia pri distribuovanom renderovaní)

GNUstep je objektovo orientovaný framework určený na vývoj bežných aplikácií. GNUstep umožňuje vývojárom rýchly vývoj softvéru poskytovaním veľkej knižnice znovu použiteľných komponentov.

Je to voľne šíriteľná implementácia NeXT-ovského OpenStep framework-u (v súčastnosti už Apple-ovského COCOA framework-u). Umožňuje vývoj programov nielen pre Unix-ové operačné systémy, ale aj pre MS Windows.

Jeho prvé riadky napísali Paul Kunz s kolegami, keď chceli preniesť program HippoDraw z NeXTSTEP-u na ostatné platformy. Namiesto prepisovania celej aplikácie a využitia jej návrhu, sa rozhodli prepísať NeXTSTEP-ovú objektovú vrstvu, na ktorej bola aplikácia závislá. Tak vznikla prvá knižnica libobjcX, ktorá im umožnila portovať HippoDraw na Unix-ový systém bežiaci s X-

Windows bez zmeny jediného riadku zdrojového kódu aplikácie. Po tom čo bol zverejnená špecifikácia OpenStep-u (1994), rozhodli sa napísať novú knižnicu pokrývajúcu toto nové API. Tento softvér pomenovali GNUstep.

GNUstep framework sa delí na **negrafickú** (GNUstep base, Foundation) a **grafickú** (GNUstep GUI, ApplicationKit) časť. GNUstep base je sada výkonných negrafických tried, umožňujúcich písanie neuveriteľného množstva nástrojov. Druhá grafická časť pokrýva vývoj kompletného grafického rozhrania aplikácie. Spolu s nástrojmi GORM a ProjectCenter sa vývoj grafického rozhrania stáva záležitosťou pár týždňov či mesiacov.

Foundation kit pokrýva tieto oblasti:

- reťazce
- kolekcie (polia, slovníky, množiny)
- práca so súbormi
- archivácia/serializácia objektov
- dátum
- distribuované objekty a medziprocesová komunikácia
- URL
- notifikácie (a distribuované notifikácie)
- vlákna
- časovače
- zámky
- výnimky
- XML

Application Kit pokrýva nasledovné oblasti:

- prvky užívateľského rozhrania
- grafika
- správa farieb (CMYK, RGB, HSB, …)
- práca s textom
- podpora dokumentov (MDI aplikácie)
- tlač
- nápoveda
- schránka
- korektor pravopisu
- drag & drop operácie

Voľba programovacieho jazyka Objective-C bol krok správnym smerom, ktorý vôbec neobmedzoval implementáciu, ba naopak ju uľahčil.

6.3 Použitie, inštalácia, rozšírenie

Kompletné funkčné strihacie jadro je rozdelené do štyroch častí, jadro, dva pluginy a demonštračný program. Jadro je vlastne knižnica obsahujúca triedy a moduly, a nazýva sa DRFoundation. Druhou časťou je sada základných efektov DreamStone. Nasleduje FFmpegPlugin rozširujúci jadro o sadu kodekov. Posledná je demonštračná aplikácia DreamMill vykonávajúca strih videa podľa zadaného programu v XML súbore.

6.3.1 Inštalácia

Inštalácia jednotlivých častí je na vlas rovnaká, preto je tu uvedený len postup inštalácie základnej knižnice. Pri inštalácii je nutné dodržať poradie inštalovaných komponent a to DRFoundation, DreamStone, FFmpegPlugin a DreamMill.

Pred samotnou inštaláciou je potrebné si zaobstarať a nainštalovať pár balíčkov, od ktorých je strihacie jadro a jeho pluginy závislé. Menovite sú to framework GNUstep (http://www.gnustep.org), od ktorého je závislé kompletné jadro a jeho pluginy, knižnica FFmpeg (http://ffmpeg.mplayerhq.hu) pre FFmpegPlugin. Ich návod na inštaláciu je možné nájsť v ich dokumentácii.

Postup inštalácie:

rozbaliť archív

```
tar -zxf DigitalReality.tar.gz
```

vstúpiť do adresára s komponentov

```
cd DigitalReality/DRFoundation
```

kompilácia

```
make
```

inštalácia

```
make install
```

6.3.2 Použitie

Použitie knižnice DRFoundation v aplikácii je veľmi jednoduché, po jej inštalácii sú všetky jej súbory (knižnica, hlavičkové súbory, …) na správnych miestach. Jej použitie pozostáva zo zahrnutia hlavičkového súboru DRFoundation/DRFoundation.h a pripojenia knižnice libDRFoundation k programu. Ukážkový GNUmakefile pre preklad našej aplikácie je možné nájsť pri programe DreamMill. Pred použitím funkcií a tried je nutné knižnicu inicializovať volaním funkcie DRFoundationInit() (inicializácia dátových štruktúr, kľúčových častí, načítanie pluginov, ...), po

skončení je nutné uvolniť alokované zdroje funkciou DRFoundationQuit(). Pracovať so strihacím jadrom je možné len medzi párom týchto dvoch funkcií.

6.3.3 Rozšírenie

Sekcia obsahujúca príklady vytvárania rozšírení, postupne sú ukázané všetky možné rozšírenia. Vytváranie rozšírení je vždy založené na odvodení zo základnej triedy a doplnením jej metód. Rozšírenia sa vytvárajú ako GNUstep projekty typu Bundle na čo treba brať ohľad pri písaní ich GNUmakefile (ukážkové GNUmakefile sa nachádza napr. pri FFmpegPlugine).

6.3.3.1 Audio/Video páska

Audio a video pásky sú vlastne dekodéry audio/video dát (streamov). Odvodzujú sa z tried a protokolov DRAudioTape a DRVideoTape, ktorých metódy musia implementovať.

```
@interface MyAudioTape : DRAudioTape <DRAudioTape>
{
// zoznam clenskych premennych
}
// zoznam metod
@end
```

Podobne je to aj s DRVideoTape. Popis významu jednotlivých metód je umiestnený v hlavičkových súboroch DRAudioTape.h a DRVideoTape.h.

6.3.3.2 Audio/Video filter

Do rozšírení spadajú aj video a audio filtre. Ako ostatné rozširujúce komponenty a pre ne platí nutnosť registrácie do systému DRRegisterVideoFilterClass([MyVideoFilter class]).

```
@interface MyVideoFilter : DRVideoFilter
{
// zoznam clenskych premennych
}
// zoznam metod
@end
```

6.3.3.3 Plugin

Vytvorenie pluginu zahrňuje v sebe vytvorenie triedy implementujúcej protokol DRPlugin.

```
@interface MyPlugin : NSObject <DRPlugin>
+ (NSString *) registerPlugin;
+ (NSString *) unregisterPlugin;
```

```
+ (NSString *) pluginName;
@end
```

Zároveň je nevyhnutné túto triedu nastaviť ako hlavnú triedu rozšírenia v GNUmakefile, a zároveň ako typ projektu určiť ako Bundle:

```
MyPlugin_PRINCIPAL_CLASS = MyPlugin
include $(GNUSTEP_MAKEFILES)/bundle.make
```

6.3.3.4 Enkodér

Rozšírenie strihacieho jadra o enkodér je tak isto jednoduché ako vytvoriť akékoľvek iné rozšírenie (takmer totožné s vytvorením filtra):

```
@interface FFmpegEncoder : DREncoder
{
// zoznam clenskych premennych
}
// zoznam metod
@end
```

Nesmieme však zabudnúť na registráciu triedy enkodéru do strihacieho jadra prostredníctvom volania funkcie DRRegisterEncoderClass([MyEncoder class]);

6.3.3.5 Loader

Loader sa odvodzuje z triedy a protokolu DRLoader, nasleduje ešte registrácia do systému DRRegisterLoader.

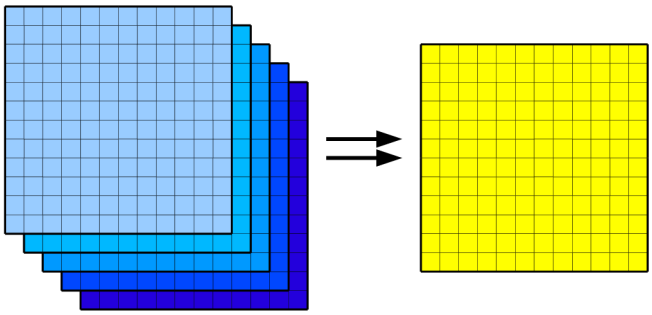
```
@interface MyLoader : DRLoader <DRLoader>
{
// zoznam clenskych premennych
}
// zoznam metod
@end
```

6.4 DreamStone – sada efektov

Návrh strihacieho jadra počíta s jeho veľkou rozšíriteľnosťou, a preto v samotnom jadre nie sú implementované niektoré funkčné časti, bez ktorých toto jadro nie je schopné pracovať, ale počíta sa s ich pridaním prostredníctvom zásuvných modulov (FFmpegPlugin, DreamStone). Plugin

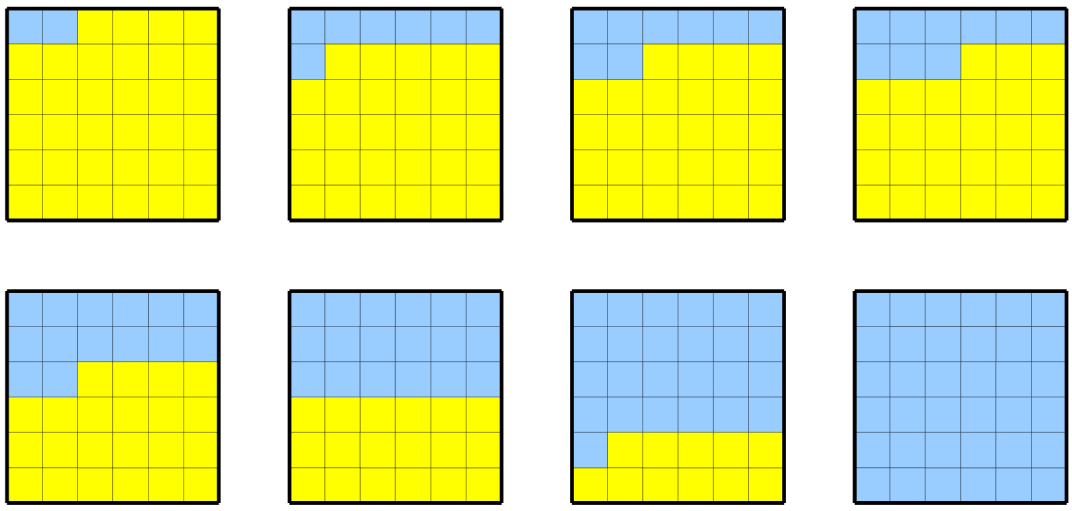
DreamStone je kolekciou základných demonštračných filtrov. Jeho inštalácia je obdobná inštalácii strihacieho jadra. Obsahuje filtre ako napr. MotionBlur a Checkboard, Echo, ...

MotionBlur – implementuje efekt, ktorý zachycuje v jednom snímku určité časové obdobie (čiže nie jeden časový okamih, ale interval). Takýto efekt je možné dosiahnuť kombinovaním určitého počtu snímkov nasledujúcich za sebou.



Obrázok 15: Filter motion blur

Checkboard – je filter šachovnicového prechodu medzi dvoma sekvenciami, na nasledovných obrázkoch je zobrazený jeho priebeh aj výsledok.



Obrázok 16: Šachovnicový filter

6.5 DreamMill – konzolový strihač

DreamMill je jednoduchá demonštračná aplikácia umožňujúca pohodlný strih. Využíva dostupné vlastnosti framework-u GNUstep ako aj strihacieho jadra, na ktorom je postavená a slúži na demonštráciu jeho funkčnosti. Jej ovládanie je jednoduché prostredníctvom konfiguračného súboru predaného ako jediný parameter programu (je to konzolová aplikácia, neobsahuje ani jeden GUI prvok). Konfiguračný súbor vychádza z formátu XML skladajúceho sa zo 4 sekcií, ich význam popisuje nasledujúca tabuľka (dôležité je dodržať poradie sekcií).

Názov sekcie	Popis významu sekcie
<tapes>	definícia audio a video pások obsahuje elementy <video_tape> a <audio_tape>
<cells>	popis renderovacích buniek popisuje audio ako aj video bunky dovolené elementy sú <audio_cell> a <video_cell>, používajúce audio/video pásky definované v sekcii <tapes>
<encoder>	sekcia parametrov predaných zvolenému enkóderu je závislá od použitého enkóderu
<sequence>	časť definujúca skladbu výslednej video sekvencie (formát, dĺžku, ...)

Nasleduje príklad konfiguračného súboru (pozor na časovanie, pri video bunkách sa udáva v nanosekundách, pri audio bunkách v samploch a je závislé na formáte sekvencie):

```
<?xml version="1.0"?>
<dream_mill_document>    <!-- uvodny tag dokumentu -->
  <tapes>                <!-- sekcia definicii pasok -->
    <!-- nacitanie video suborov a ich rozdelenie a pomenovanie
         na audio a video pasky -->
    <tape filename="apocalypto_clip1.avi">
      <video_stream name="v1"/>
      <audio_stream name="a1"/>
    </tape>
    <tape filename="apocalypto_clip2.avi">
      <video_stream name="v2"/>
      <audio_stream name="a2"/>
    </tape>
    <!-- nacitanie audia vo formate mp3 -->
    <tape filename="joseph1.mp3">
      <audio_stream name="a1"/>
```

```
    </tape>
    <tape filename="phantom1.mp3">
      <audio_stream name="a2"/>
    </tape>
  </tapes>
  <cells>               <!-- definicie buniek -->
    <!-- vytvorenie video bunky zalozenej na paske v1 -->
    <video_cell name="v1" tape_name="v1">
      <filters>
       <filter filter_name="Checkboard"/>
      </filters>
    </video_cell>
    <video_cell name="v2" tape_name="v2">
      <filters>
       <filter filter_name="Checkboard"/>
      </filters>
    </video_cell>
    <!-- vytvorenie audio bunky zalozenej na audio paske a1 -->
    <audio_cell name="a1" tape_name="a1">
    </audio_cell>
    <audio_cell name="a2" tape_name="a1">
    </audio_cell>
  </cells>
  <!-- nastavenie enkoderu -->
  <encoder encoder_name="FFmpegEncoder" audio_bit_rate="64000"
    sample_rate="44100" video_bit_rate="4000000"
    output_file="strih1.avi">
  </encoder>
  <!-- parametre sekvencie -->
  <sequence name="strih1" start="0" end="15000000" width="640"
    height="352" fps="24" sample_rate="44100" channels="2">
    <!-- umiestnenie buniek do vyslednej sekvencie -->
    <video_cell cell_name="v1" position="0" track="0"/>
    <audio_cell cell_name="a1" sample_position="0" track="0"/>
    <audio_cell cell_name="a2" sample_position="44100" track="1"/>
  </sequence>
</dream_mill_document>
```

6.6 FFmpegPlugin – podpora knižnice FFmpeg

FFmpegPlugin rozširuje strihacie jadro DRFoundation o sadu enkodérov/dekodérov poskytovaných knižnicou FFmpeg. Tento plugin nepokrýva všetky možnosti poskytované knižnicou FFmpeg, skôr slúži na demonštráciu ako vzorová implementácia podpory kodekov.

Obsahuje päť tried, zahrňujúcich podporu kodekov z knižnice FFmpeg. Na dosiahnutie tejto podpory implementuje štyri triedy:

- FFmpegPlugin
- FFmpegVideoTape
- FFmpegAudioTape
- FFmpegLoader
- FFmpegEncoder

FFmpegPlugin plní len pomocnú funkciu pri načítaní pluginu, v podstate len zaregistruje ostatné triedy do systému.

FFmpegVideoTape v sebe obsahuje podporu všetkých video dekodérov knižnice FFmpeg, jeho výstupným formátom je formát RGBA

FFmpegAudioTape je obdoba video pásky FFmpegVideoTape, ale pre prácu s audiom.

FFmpegLoader zastupuje podporu kontajnerových formátov knižnice FFmpeg, jeho princíp je nasledovný, získaný video súbor rozdelí na audio/video pásky, tieto pásky ako objekty DRVideoTape a DRAudioTape predá ďalej jadru.

FFmpegEncoder prijíma vyrenderované audio sample a video obrázky a kóduje ich do video súboru, pričom používa sadu enkodérov z knižnice FFmpeg.

7 Záver

Cieľom tejto práce bolo preskúmať možnosti súčasných nástrojov používaných pre strih videa a pomocou získaných poznatkov navrhnúť jadro pre strih video sekvencií. Vytýčené ciele sa tejto práci podarilo splniť. Boli preskúmané možnosti aplikácií z rôznych oblastí, od veľkých strihacích štúdií až po malé jednoúčelové utility.

Hlavnou úlohou tejto práce bolo navrhnutie a implementácia modulárneho jadra pre strih video sekvencií, s veľkým dôrazom kladeným na rozšíriteľnosť a univerzálnosť. Univerzálnosť umožňuje jeho použitie v rôznych typoch aplikácií, rozšíriteľnosť otvára dvere novým kodekom, audio/video filtrom, ...

Implementačne je jadro založené na v súčasnosti v svete osobných počítačov (Windows, Linux) neznámom prostredí GNUstep a programovacom jazyku Objective-C (vo svete Mac OS X sú tieto pojmy veľmi dobre známe, pretože sú to jeho natívne vývojové nástroje), čím rozširuje ponuku preň dostupných aplikácií a knižníc.

Pokračovať v tejto práci sa dá viacerými spôsobmi. Jedným z nich je lepšia a komplexnejšia implementácia FFmpegPluginu, príp. väčšia sada základných filtrov. Inou oblasťou je optimalizácia strihacieho jadra a jeho súčastí na dosiahnutie vyššieho výkonu. To v sebe zahrňuje predovšetkým optimalizáciu filtrov na úrovní asembleru (využitie sád inštrukcií MMX, SSE, SSE2, …). Ďalšia prázdna kapitola sú obrazové formáty (RGBA je implementovaný, ale napr. podpora YCC formátov určite nie je krok vedľa). Najlepším pokračovaním tejto práce je použitie a rozšírenie tohto strihacieho jadra na vytvorenie kompletného strihacieho štúdia.

Literatúra

[1] prispievatelia Wikipédia, *Audio Video Interleave* [online], Wikipédia, 2007, [cit. 2007-05-05]. Dostupné z: http://en.wikipedia.org/wiki/Audio_Video_Interleave

[2] prispievatelia Wikipédia, *Codec* [online], Wikipédia, 2007, [cit. 2007-05-05]. Dostupné z: http://en.wikipedia.org/wiki/Codec

[3] *FFmpeg documentation* [online], Wikipédia, 2007, [cit. 2007-05-05]. Dostupné z: http://ffmpeg.mplayerhq.hu/documentation.html

[4] Pavel Tišnovský, *JPEG-král rastrových grafických formátů?* [online], 2006, [cit. 2007-05-05]. Dostupné z: http://www.root.cz/clanky/jpeg-kral-rastrovych-grafickych-formatu/

[5] prispievatelia Wikipédia, *Kodek* [online], Wikipédia, 2007, [cit. 2007-05-05]. Dostupné z: http://sk.wikipedia.org/wiki/Kodek

[6] prispievatelia Wikipédia, *Linear video editing* [online], Wikipédia, 2007, [cit. 2007-05-05]. Dostupné z: http://en.wikipedia.org/wiki/Linear_video_editing

[7] *LiVES* [online], 2007, [cit. 2007-05-05]. Dostupné z: http://lives.sourceforge.net/

[8] prispievatelia Wikipédia, *Non-linear editing system* [online], Wikipedia, 2007,[cit.2007-05-05]. Dostupné z: http://en.wikipedia.org/wiki/Non-linear_editing_system

[9] prispievatelia Wikipédia, *Sony Vegas* [online], Wikipédia, 2007, [cit. 2007-05-05]. Dostupné z: http://en.wikipedia.org/wiki/Sony_Vegas

[10] prispievatelia Wikipédia, *Video for Windows* [online], Wikipédia, 2007, [cit. 2007-05-05]. Dostupné z: http://en.wikipedia.org/wiki/Video_for_Windows

[11] *VirtualDub* [online], 2007, [cit. 2007-05-05]. Dostupné z: http://www.virtualdub.org/

[12] prispievatelia Wikipédia, *VirtualDub* [online], Wikipédia, 2007, [cit. 2007-05-05]. Dostupné z: http://en.wikipedia.org/wiki/Virtualdub

[13] GNUstep dokumentácia, [cit. 2007-05-05]. Dostupné z: http://www.gnustep.org/

Zoznam príloh

Príloha 1. CD so zdrojovými kódmi, demonštračnou aplikáciou