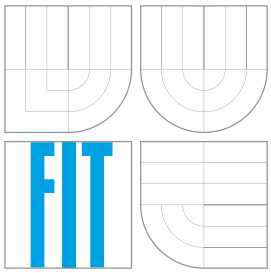
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

GUI KE STAVBĚ GRAFU FILTRŮ PRO FFMPEG

GUI TO FFMPEG FOR BUILDING A FILTER GRAPH

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. DANIEL KLIMAJ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. DAVID BAŘINA

BRNO 2016

Abstrakt

Táto práca sa zaoberá implementáciou aplikácie s grafickým používateľským rozhraním pre tvorbu grafov filtrov v FFmpeg. Práca obsahuje návrh riešenia, ktorý obsahuje návrh grafického rozhrania a návrh objektového modelu reprezentácie grafov filtrov, a rovnako aj popis implementácie zhotovenej na základe týchto návrhov. Súčasťou práce je porovnanie dosiahnutého výsledku s existujúcimi riešeniami.

Abstract

This work deals with implementation of applications with a graphical user interface for creating filter graph in FFmpeg. The work includes a design of a solution which contains a design of a graphical user interface and object model representation of the filter graph, as well as a description of the implementation zhotovenje on the basis of these designs. Part of this work is a comparison of the achieved result with existing solutions.

Kľúčové slová

graf filtrov, GUI, video, obraz, filtrovanie, FFmpeg

Keywords

filter graph, GUI, video, image, filtering, FFmpeg

Citácia

KLIMAJ, Daniel. *GUI ke stavbě grafu filtrů pro FFmpeg.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Bařina David.

GUI ke stavbě grafu filtrů pro FFmpeg

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Davida Bařinu. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Daniel Klimaj
17. mája 2016

Poďakovanie

Ďakujem vedúcemu práce Ing. Davidovi Bařinovi za trpezlivosť a cenné rady pri tvorbe tejto práce.

Obsah

Úvod

Video na rôznych nosičoch a v rôznych formátoch je tu už s nami vyše storočia a za túto pomerne dlhú dobu ušlo poriadny kus cesty. V poslednej dekáde 19-teho storočia. sme tu mali len pár sekúnd dlhé filmy, ktoré neobsahovali žiaden zvuk. Postupne pribudol zvuk, animácie, z čierno-bieleho obrazu sa stal farebný a vzrastala dĺžka filmov, ale pojem digitálne video bola stále veľká neznáma.

To sa zmenilo až na prelome 70-tych a 80-tych rokov minulého storočia, kedy sa začali objavovať prvé nástroje pracujúce s digitálnym videom a vznikol prvý formát digitálneho videa známy ako H.120. Ten mal maximálne rozlíšenie len 176x144 pixelov a prenosovú rýchlosť dva megabity za sekundu. To sa z dnešného pohľadu môže zdať až ukrutne málo, ale je to možno práve tento formát, ktorému vďačíme za to, že tu dnes máme formáty s rozlíšením niekoľko tisíc ku niekoľko tisícom pixelov s prenosovými rýchlosťami niekoľko megabajtov či dokonca gigabajtov za sekundu. Od vzniku prvého formátu sa dodnes objavil nespočet nových formátov, niektoré sa ujali, na ďalšie sa zabudlo.

K pojmu video neodmysliteľne patrí aj pojem video filter. Dnes tu máme obrovský počet filtrov, či už je ich účel odstránenie šumu, zmena kontrastu, zmena rozlíšenia alebo niečo úplne iné. V praxi sa môžeme stretnúť sa prípadom keď niekoľko filtrov postupne spracuje vstupné video na určitý výstup, takto zreťazené filtre sa nazývajú graf filtrov. Vo svete počítačov sa postupne objavilo niekoľko knižníc a nástrojov poskytujúce prácu s video filtrami a grafmi filtrov, medzi tie najznámejšie určite patrí FFmpeg.

Práve vývoj aplikácie pre tvorbu grafov filtrov v FFmpegu je hlavným účelom tejto práce. Okrem toho práca popisuje základné princípy a spôsoby práce s multimediálnym frameworkom FFmpeg. Získané informácie sú potom použité pri návrhu a implementácii požadovanej aplikácie.

Dokument pozostáva z troch kapitol. Prvá kapitola vysvetľuje dôležité pojmy, o ktoré sa oprierajú nasledujúce časti a taktiež poskytuje zoznámenie sa s najznámejšími multimediálnymi frameworkmi pracujúcimi s grafom filtrov. Druhá kapitola na úvod popisuje výber nástroja použitého k implementácii grafického rozhrania. Ďalej sa venuje návrhu riešení dôležitých problémov, akými sú spôsob ukladania dát alebo reprezentácia grafu filtrov vo vyvíjanej aplikácii. V druhej polovici sa kapitola venuje popisu vlastnej implementácie aplikácie - implementovaných tried, ich význam a spôsob vzájomnej komunikácie a interakcie. V záverečnej tretej kapitole je vytvorená aplikácia porovnaná s existujúcimi riešeniami.

Kapitola 1

Vymedzenie pojmov a rozbor technológií

Táto kapitola popisuje technológie a pojmy, ktorých znalosť je potrebná pre správne pochopenie princípov reprezentácie farieb, uloženia videí a ich filtrovania. Na úvod kapitola popisuje význam dôležitých pojmov používaných v oblasti spracovania obrazu a videa, a potom predstaví rôzne technológie pracujúce s multimédiami a grafmi filtrov.

1.1 Farba, pixel, farebné modely

Farba je vnem viditeľnej časti spektra – vlnové dĺžky od približne 400 nanometrov do približne 700 nanometrov – dopadajúcej na sietnicu oka, ktorá obsahuje dva typy svetlocitlivých receptorov – čapíky a tyčinky. Tyčinky sú citlivejšie ako čapíky, ovplyvňujú vnímanie jasu a umožňujú nočné videnie. Čapíky sú troch druhov, z ktorých je každý citlivý na svetlo inej vlnovej dĺžky - najcitlivejšie sú na svetlo modrej, červenej a zelenej časti spektra. Z toho dôvodu sú hlavnými farbami monitorov červená (R, red), zelená (G, green) a modrá (B, blue).

Farebný model je matematický model popisujúci farbu. Vyššie spomínaná trojica farieb tvorí farebný model RGB, v ktorom sú R, G a B nezávislými zložkami s hodnotami 0 až MAX. Maximálna hodnota zložky závisí od počtu bitov, v ktorých je informácia uložená. Farebný priestor udáva každej tejto zložke farebnosť, najčastejšie v chromatickom diagrame CIExy. Rozsah farieb, ktorý dokáže konkrétny farebný priestor pokryť je udávaný trojuholníkom medzi vrcholmi zložiek R, G a B, sa nazýva gamut. Ak je obrázok reprezentovný pixelmi v modely RGB a nie je k nemu informácia o farebnosti (chromatickosti) jednotlivých zložiek R, G a B, nebude sa dať verne zobraziť alebo vytlačiť [1].
V rastrovej grafike sú spracované a zobrazované dáta ukladané diskrétne vo forme rastrovej matice. Jeden prvok tejto matice sa nazýva **pixel** [12]. Farebný model YC_bC_r je transformáciou farebného modelu RGB, ktorý ma niekoľko možných variánt v závislosti od použitej normy alebo vstupných a výstupných hodnôt.

Pod pojmom **formát pixelu** sa rozumie použitý farebný model a spôsob reprezentácie jeho zložiek v pamäti. Napríklad v prípade farebného modelu RGB s 8 bitmi pre každý kanálmôžu byť pixely uložené v modely RGB24 alebo BGR24 podľa poradia jednotlivých bajtov v pamäti. Modely ARGB, RGBA alebo RGBA32 pridávajú dodatočný bajt reprezentujúci alfakanál (priehľadnosť).

Pre značenie formátu pixelu sa využíva mnoho vzájomne mierne odlišných spôsobov

zápisu. Napríklad v prípade formátu YUV444 resp. YC_bC_r444 to znamená, že je pixel reprezentovaný troma bajtmi v nepodvzorkovanom formáte YUV resp. YC_bC_r. Ak je využité podvzorkovanie, tak niektoré pixely tvoria bloky so spoločnou niektorou farebnou zložkou C_b alebo C_r. Takéto pixely so spoločnou farebnou zložkou dokopy tvoria **makropixel**. Typicky sa využívajú bloky o výške dvoch a o šírke štyroch pixelov a podvzorkovanie sa zapisuje ako trojica čísel vo formáte *J:a:b*. *J* udáva šírku bloku (spravidla J=4), *a* udáva počet farebnostných zložiek horného riadku a *b* udáva počet farebnostných zložiek spodného riadku. Bežne používanými typmi podvzorkovania sú 4:4:4 (nepodvzorkované), 4:2:2 (farebnostné vzorky horizontálne podvzorkované na polovicu) alebo 4:1:1 (farebnostné zložky horizontálne podvzorkované na štvrtinu) [1].

1.2 Obraz, reprezentácia obrazu

Na farebný **obraz** sa možno pozerať ako na dvojrozmernú diskrétnu funkciu s parametrami x a y, udávajúce súradnice bodu, ktorej hodnotou je farba obrazového bodu. V počítačoch sa bežne používa farebný model RGB a v televízoroch sa bežne využívajú rôzne varianty modelu YUV. Z toho dôvodu je pre stratovú konverziu vhodné použiť práve niektorý z derivátov modelu YUV. Farebný model YUV je farebný model YC_bC_r, kde zložka U značí zložku C_b a zložka V značí zložku C_r. Poradie a veľkosť komponent je udávaná v rámci makropixelu.

Vo farebnom modely YUV je farba udávaná ako trojica *(y, u, v)*. Oddelením jasu (zložka Y) od farebných informácií, na ktoré je ľudské oko menej citlivé, sa môže na farebných zložkách dopsutiť vyššej straty informácií, čo vedie k lepšiemu pomeru kvality ku veľkosti skomprimovaného obrazu. Základným farebným modelom pre prácu s obrazom je farebný model RGB s jedným bajtom pre každú zložku s hodnotami v itervale $\langle 0; 255 \rangle$. Pôvodné vzťahy pre prevod RGB do YC_bC_r využívali reálne čísla, čo viedlo ku stratám pri spätnom prevode. Ak je potrebné, aby pri prevode nedošlo ku strate informácií, musí sa použiť niektorý z modelov, u ktorého je zaručená bezstratová reverzibilita prevodu.

1.3 Kompresia a kódovanie obrazových dát

Pod pojmom **kompresia dát** rozumieme postup, pomocou ktorého dosiahneme zmenšenie objemu alebo dátového toku dát. Určitá forma kompresie sa vyskytuje viac-menej vo všetkých multimediálnych formátoch (napríklad BMP, PNG, JPEG, MPEG-4, atď.). Kompresia môže byť bezstratová alebo stratová. Dekompresia je postup, v ktorom dochádza k pokusu o zrekonštruovanie komprimovaných dát.

Bezstratová kompresia zmenšuje redundanciu vstupných dát a pôvodné dáta sa dajú zrekonštruovať bez skreslenia. *Stratová kompresia* odstraňuje nepodstatné dáta z pohľadu vnímania človekom (viď 1.1). Pri použití stratovej kompresie dochádza k nenávratnej strate niektorých informácií, a tak pôvodné dáta nejde presne zrekonštruovať. Dochádza ku skresleniu dekomprimovaného signálu, ale z pohľadu ľudského vnímania môže byť táto strata nepostrehnuteľná. Stratová kompresia dosahuje podstatne vyšších kompresných pomerov. Účinnosť kompresnej metódy možno zmerať pomocou kompresného pomeru, čo je podiel medzi veľkosťou komprimovaných dát a veľkosťou nekomprimovaných dát. Ak je tento pomer menší ako 1, tak sa jedná o kompresiu dát, v opačnom prípade sa jedná o expanziu dát.

Kódovanie je proces vzájomne jednoznačného priradenia symbolov jednej abecedy

symbolom druhej abecedy. Kódy môžu byť priradené jednotlivým symbolom alebo blokom symbolov vstupného toku. Kódovanie, ktorého následkom je zníženie redundancie dát možno nazývať kompresiou. Kódovanie a dekódovanie vykonáva kóder a dekóder [1].

1.4 Video, snímka, kodek

Pod pojmom **digitálne video** možno chápať diskrétny signál troch premenných *(x, y, t)* - priestorové súradnice x a y, a čas t - kde hodnotou každého vzorku takéhoto signálu je farba. Farba býva bežne vyjadrená ako trojica (R, G, B) alebo (Y, C_b, C_r). Z dôvodu obrovskej pamäťovej náročnosti pri práci s nekomprimovanou formou videa sa s videom pracuje ako so sekvenciou snímok.

Základnou jednotkou videa je **snímka** (frame). S videom sa potom pracuje ako so sekvenciou snímok, a to z dôvodu extrémnej pamäťovej náročnosti nekomprimovanej formy signálu. Každý multimediálny framework, aplikácia a formát pracuje predovšetkým (alebo výhradne) so snímkami.

Video formát udáva spôsob uloženia videa na disku. Formát môže byť nekomprimovaný, stratový a bezstratový. Video formát je špecifikácia komprimovaných dát, ktorá môže byť štandardizovaná niektorou zo štandardizačných organizácií (ITU-T, ISO/IEC).

Video kodek je kombinácia kóderu a dekóderu - hardware alebo software, ktorý je schopný zakódovať a lebo dekódovať dátový prúd do alebo z určitého formátu. Video kodeky sú konkrétne implementácie video formátu. Dátové prúdy sú uložené v kontajnerových formátoch spolu s ďalšími informáciami (titulky, kapitoly, atď.). Spolu s audiom a ďalšími informáciami sú tieto prúdy uložené v multimediálnych kontajneroch [1].

1.4.1 Vlastnosti video snímkov

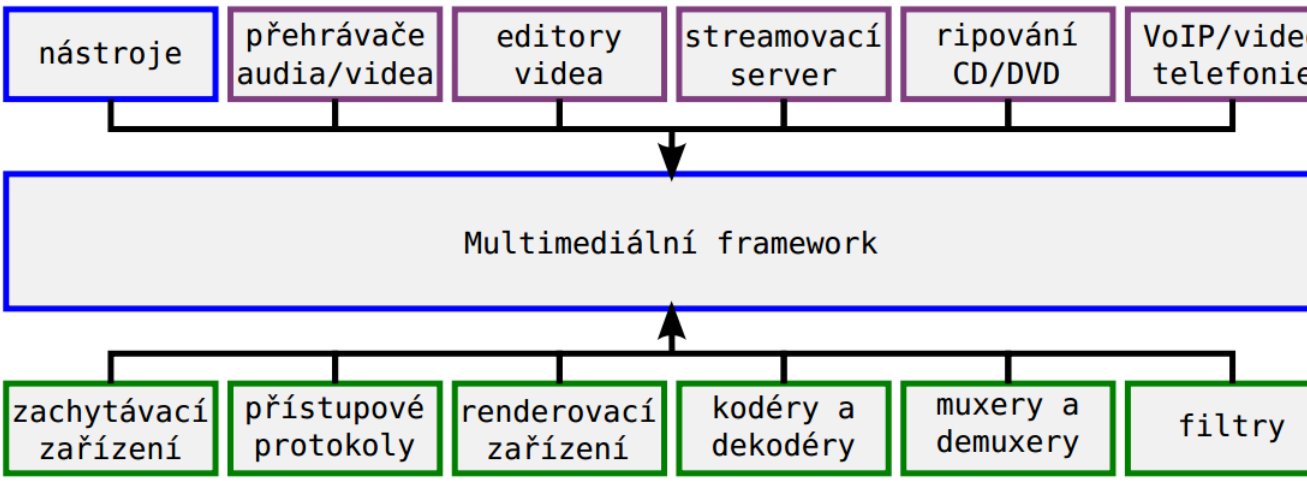
Prenosová rýchlosť alebo dátový tok (bitrate) udáva počet bitov komprimovaného videa potrebných k jeho prehraniu za sekundu - bps (bits per second), kbps, Mbps. Ak sa táto rýchlosť po celú dobu trvania vido stopy nemení, hovoríme o konštantnej prenosovej rýchlosti - *constant bitrate* (CBR). V prípade konštantnej prenosovej rýchlosti dochádza ku zmenám kvality videa počas jeho priebehu pretože náročnejšie scény vyžadujú vyšší dátový tok, u menej náročných scén naopak môže dochádzať k plytvaniu dátového toku. V prípade, keď sa prenosová rýchlosť počas priebehu videa mení, hovoríme o premennej prenosovej rýchlosti - *variable bitrate* (VBR). Špeciálnym typom premennej prenosovej rýchlosti je priemerná prenosová rýchlosť - *average bitrate* (ABR). Priemerná prenosová rýchlosť sa snaží dosiahnuť dlhodobo rovnakého priemerného dátového toku, vyžaduje ale viacnásobný prechod kompresnou metódou.

Snímková rýchlost (framerate, frames per second - FPS) udáva rýchlosť, akou sú snímky zobrazované pri prehrávaní. Jednotkou snímkovej rýchlosti je Hz, ale často sa nesprávne používa jednotka *fps*. Najčastejšie hodnoty sú 24 (film), 25 (TV) a 60 (HDTV).

Snímky nemusia byť v kontajneroch uložené v poradí za sebou. Preto musia so sebou niesť informáciu o svojom poradí - časovými razítkami (timestamps). Týmito razítkami sú **Decoding timestamp** (DTS) a **Presentation timestamp** (PTS). Presentation timestamp udáva poradie, v akom majú byť snímky pri prehrávaní zobrazované a Decoding timestamp poradie, v akom majú byť snímky dekódované [1].

1.5 Multimediálny framework a video filter

Multimediálny framework je súbor knižníc a nástrojov pre prácu s multimediálnymi dátami, najčastejšie audiom a videom. Multimediálny framework poskytuje rôzne služby ako napríklad otvorenie súboru, kompresiu alebo dekompresiu snímok, ukladanie do kontajnerov, atď. Frameworky môžu mať v sebe pevne daný graf tokov ako napríklad multimediálny framework Video for Windows alebo je možné graf toku ovplyvniť ako napríklad vo frameworkoch GSteamer, DirectShow alebo Media Foundation [1].



Obr. 1.1: Multimediálny framework (modré ohraničenie) a jeho vzťah k aplikáciám (fialové) a zásuvným modulom (zelené) (Zdroj: [1])

Filter - V kontexte multimediálnych frameworkov možno filtre rozdeliť do troch skupín:

1. zdroje dátového toku
2. transformačné filtre (kódery, dekódery, atď.)
3. cieľové filtre

Filtre sú navzájom prepojené prostredníctvom vstupov a výstupov, v závislosti od zvoleného multimediálneho frameworku sa tieto miesta nazávajú padmi, pinmi alebo sú prípadne pomenované inak. Podľa svojho typu môže mať filter žiadny, jeden alebo viacej vstupov alebo výstupov. Aby bolo spojenie filtrov možné, musí sa zhodovať typ dátového toku na výstupe jedného filtru s typom dátového toku na vstupe druhého filtru. Nemožno napríklad napojiť výstup dekóderu videa na vstup zvukovej karty. V niektorých prípadoch môže byť multimediálny framework schopný do určitej miery výstupný tok skonvertovať. Napríklad môže zmeniť vzorkovaciu frekvenciu alebo formátu pixelu. Vzájomne prepojené filtre prostredníctvom pinov alebo padov tvoria **graf filtrov**. V niektorých multimediálnych frameworkoch sa miesto označenia graf filtrov používa označenie *pipeline* [1].

1.6 Prehľad multimediálnych frameworkov

Táto časť poskytuje prehľad niektorých bežne používaných multimediálnych frameworkov. Konkrétne sú to DirectShow, Media Foundation, GStreamer a Phonon. Multimediálny framework FFmpeg, na ktorý sa táto práca zameriava nie je v tejto časti popísaný, ale je

mu venovaná nasledovná časť. Účelom tejto časti je predovšetkým poukázať na rozdiely v používanej terminológii v jednotlivých multimediálnych frameworkoch.

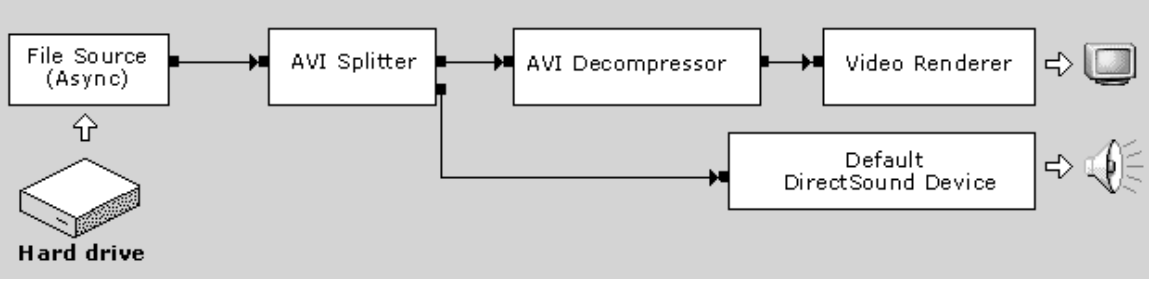
1.6.1 DirectShow

DirectShow je framework vyvinutý firmou Microsoft ako nástupca multimediálneho frameworku Video for Windows. Je založený na objektovom modely Component Object Model (COM). DirectShow poskytuje sadu štandardných filtrov a rovnako umožňuje aj pridávanie nových filtrov.

Filtre môžu byť v jednom z troch stavov – bežiaci, zastavená a pozastavený – a možno ich rozdeliť do nasledujúcich skupín [6]:

- **Source** – zdrojové filtre, ktoré sú vstupným bodom toku dát do grafu, zdrojovými filtrami sú napríklad multimediálny súbor, kamera, atď.
- **Transform** – prijíma vstupný tok dát, spracuje ich a vytvorí výstupný tok dát, príkladom takýchto filtrov sú enkóder, dekóder a iné.
- **Render** – je výstupným filtrom pre dátový tok, prijímajú dáta a predstavujú ich používateľovi, a to buď zápisom do súboru, výstupom na zobrazovacie zariadenie alebo podobným spôsobom.
- **Splitter** – rozdeľuje vstupný tok na dva a viac výstupných tokov, bežne pri tom dochádza k nejakému spracovaniu vstupného toku, príkladom takéhoto filtru je AVI Splitter, ktorý vstupný tok rozdeľuje na video a audio stream (tok).
- **Mux** – pracuje opačne ako splitter, na vstupe má niekoľko tokov a vytvára z nich jediný tok.

Výhodami oproti predchodcovi Video for Windows sú graf filtrov alebo možnosť automatickej konverzie farebných modelov. Je spätne kompatibilný s Video for Windows a obsahuje jeho kodeky z Video Compression Manager obalené transformačným filtrom AVI Decompressor alebo AVI Compressor. Pre využívanie je treba nainštalovať Windows SDK (pôvodne DirectX SDK), ktoré okrem DirectShow obsahuje aj editačný nástroj *GraphEdit* umožňujúci vizualizovať graf filtrov a jednoduchým spôsobom ho upravovať pomocou myši. Dnes je zo strany Microsoftu považovaný za zastaralý a od Windows Vista je nahradený multimediálnym frameworkom Media Foundation [1].



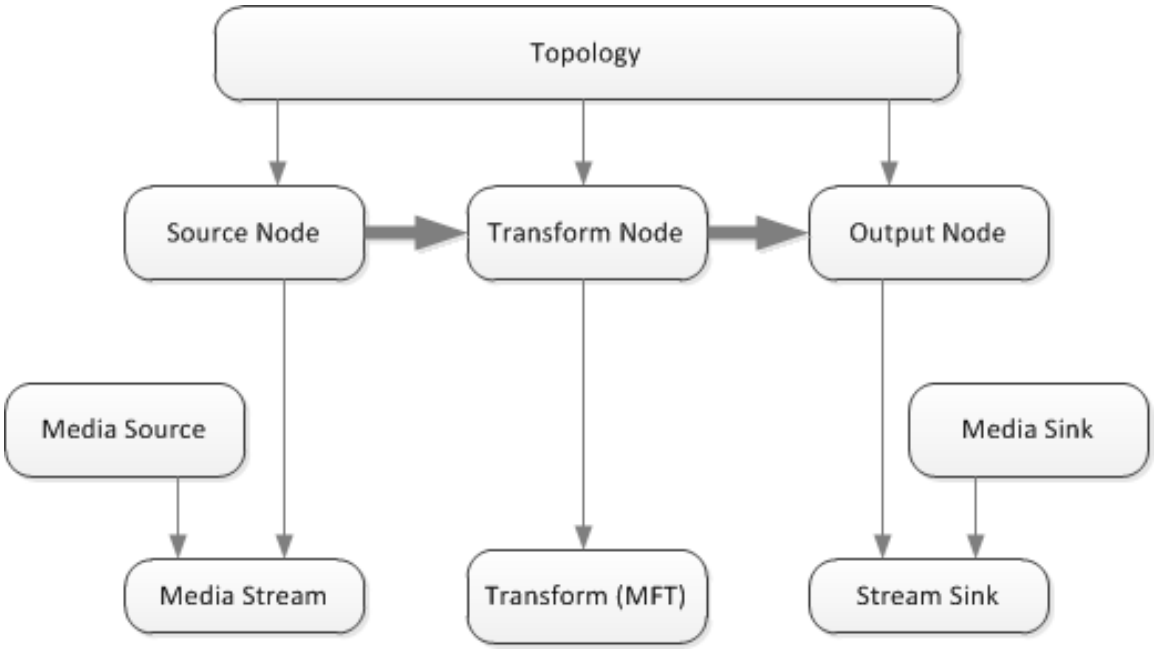
Obr. 1.2: Spracovanie AVI súboru v DirectShow (Zdroj: [6])

1.6.2 Media Foundation

Media Foundation je multimediálny framework od Microsoftu vyvinutý ako nástupca multimediálneho frameworku DirectShow pre moderné verzie MS Windows počínajúc s Windows Vista. Podobne ako DirectShow využíva COM model. Vylepšeniami oproti DirectShow sú väčšia podpora formátov, zjednodušenie programovacieho modelu, podpora hardwarových zariadení v pipeline alebo zameranie na HD (high-definition) obsah. Je súčasťou aktuálnej verzie Windows SDK [7]. Graf filtrov sa v Media Foundation označuje ako topológia (topology), filtre sa zas označujú ako uzly (nodes) a tok dát je reprezentovaný prepojením jednotlivých uzlov. Media Foundation rozlišuje štyri typy uzlov:

- **Source** predstavuje tok dát na zdrojovom multimediálnom nosoči
- **Transform** predstavuje Media Foundation transformáciu (MFT), v skratke si pod týmto pojmom môžeme predstaviť akýkoľvek uzol ležiaci medzi zdrojovým a výstupným uzlom.
- **Output** predstavuje výstupný bod pre dátový tok.
- **Tee** je špeciálnym typom uzlu, ktorý nepredstavuje element topológie, ale rozdeľuje tok dát viacerými smermi.

Grafický nástroj pre prácu s topológiami je *TopoEdit*. TopoEdit je súčasťou aktuálneho balíka Windows SDK.



Obr. 1.3: Topológia v Microsoft Foundation (Zroj: [7])

1.6.3 GStreamer

GStreamer je open source multimediálny framework využívaný v aplikáciách ako Amarok, Banshee, Kaffeine, atď. Je založený na grafe filtrov, ktorý sa v jeho prípade nazýva

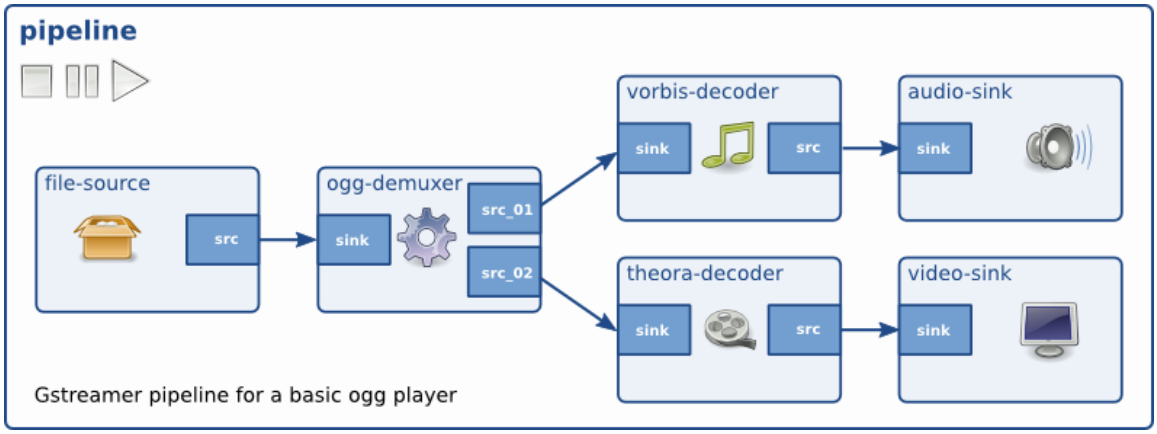
pipeline [1]. Základnými stavebnými prvkami grafu filtrov (pipeline) sú element, pad a bin. Element je v prípade GStreameru synonymum pre filter a podobne ako v predchádzajúcich spomínaných frameworkoch sa delia do niekoľkých skupín:

- **Source** – sú vstupný bod pre dáta do pipeliny, napríklad načítaním dát zo súboru.
- **Filters, convertors, demuxers, muxers and codecs** – filtre majú práve jeden vstupný a práve jeden výstupný dátový tok a vykonávajú nejaké spracovanie dátového toku, ostatné majú jeden alebo viac vstupných alebo výstupných tokov, napríklad demuxer má niekoľko vstupov a na výstupe jediný tok.
- **Sink** – sú koncové bod pre tok dát v grafe, môžu napríklad zapísať dátový tok do súboru.

Spojením niekoľkých elementov vzniká pipeline, ktorá musí obsahovať minimálne source a sink element. Pad je rozhranie elementu, pomocou ktorého komunikuje s okolím. Je definovaný svojím smerom a dostupnosťou. Smer je určený z pohľadu elementu na základe smeru toku dát. Elementy prijímajú dáta cez tzv. sink pad a vytvárajú dátový tok na tzv. source padoch. Elementy nemusia mať pri vytvorení vytvorené všetky pady, čas v ktorom sa pad vytvorí je daný ich dostupnosťou, ktorá môže byť:

- **always** – tieto pady existujú stále a sú vytvorené spolu s elementom
- **sometimes** – exisujú len nejakú dobu a môžu prípadne zaniknúť
- **on demand** – vzniknú len v prípade, keď sú o to explicitne požiadané aplikáciou.

Bin je kontajner pre elementy a sám o sebe je tiež elementom, takže sa s ním môže pracovať ako s elementom. Pipeline je špeciálny kontajner (bin) najvyššej úrovne [4].

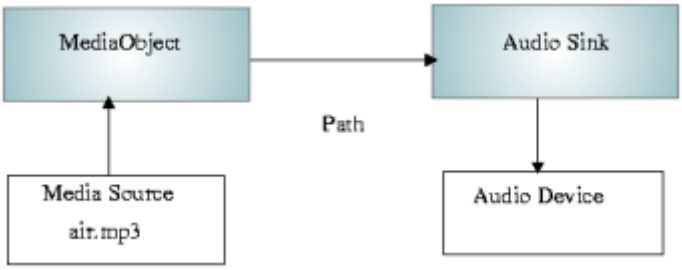


Obr. 1.4: Pipeline v GStreameri (Zdroj: [4])

1.6.4 Phonon

Phonon je multimediálny framework štandardne používaný v KDE alebo Qt aplikáciách (súčasťou Qt od verzie 4.4, ale v Qt5 bol nahradený s QtMultimedia). Podporuje niekoľko backendov, mimo iné GStreamer, DirectShow alebo QuickTime. Graf filtrov v Phonon tvoria základné elementy MediaObject, Sink a Path. MediaObject poskytuje základnú kontrolu

nad mediálnym tokom (media stream) – spustenie, pozastavenie a zastavenie. Dáta sú do grafu dodávané pomocou inštancií triedy MediaSource, ktoré ale nie sú súčasťou grafu. MediaObject sám o sebe nedokáže upravovať bajtový prúd. K tomuto účelu slúžia špeciálne uzly – procesori (Processor) umiestnené kdekoľvek medzi objektami MediaObject a Sink. Sink je výstupným uzlom grafu, najčastejšie sa jedná o renderovacie zariadenie. Na rozdiel od MediaObject má len limitovanú kontrolu nad dátovým tokom (napríklad ovládanie hlasitosti u audio streamov). Path – cesta – predstavuje spojenie medzi objektami grafu [9].



Obr. 1.5: Graf filtrov audio streamu v Phonon (Zdroj: [9])

1.6.5 Zhrnutie multimediálnych frameworkov

Ako možno vidieť na prípadoch uvedených multimediálnych frameworkoch, nie je nijak presne dané, ako označovať grafy filtrov, filtre alebo spôsoby prepojenia jednotlivých filtrov. Možno preto má každý zo spomínaných multimediálnych frameworkov iné pomenovanie spomínaných elementov – pipeline, topology, pin, pad, atď.. Preto je na mieste si určiť spôsob označovania týchto elementov. Ďalej v tejto práci sa bude graf filtrov označovať ako pipeline, filter zostane filter a miesta prepojenia filtrov budú pomenované ako pin.

1.7 FFmpeg

Čo predstavuje FFmpeg asi najlepšie vystihuje nasledujúci odstavec z oficiálnej stránky projektu: *„FFmpeg je vedúci multimediálny framework schopný dekódovať, zakódovať, prekódovať, multiplexovať* [1] *, demultiplexovať* [2] *, streamovať, filtrovať a prehrávať v podstate všetko čo človek a stroje vytvorili. Podporuje väčšinu podivných prehistorických formátov až po tie najšpičkovejšie. Nezáleží na tom či boli navrhnuté štandardovými výbormi, komunitou alebo korporáciou. Taktiež je veľmi prenosný" [3].*

FFmpeg je tvorený niekoľkými knižnicami:

- **libavcodec** – poskytuje základný framework pre kódovanie a dekódovanie, obsahuje množstvo kóderov a dekóderov pre audio, video a titulky.
- **libavutil** – knižnica pre podboru multiplatformového programovania, obsahuje prenosné reťazcové funkcie, generátory náhodných čísel, dátové štruktúry, matematické

[1] Multiplexovanie (angl. multiplexing, skr. muxing) – metóda zlúčenia viacerých dátových tokov do jedného [2]

[2] Demultiplexovanie (angl. demultiplexing, skr. demuxing) – metóda rozdelenia jedného dátového toku na viac tokov [2]

funkcie, kryptografické funkcie alebo funkcionality potrebné pre prácu s multimédia (enumerátory, atď.).

- **libavformat** – poskytuje framework pre multiplexovanie a demultiplexovanie dátových tokov, obsahuje niekoľko multiplexorov a demultiplexorov pre rôzne formáty.
- **libavfilter** – obsahuje množstvo filtrov a poskytuje potrebnú funkcionalitu pre filtrovanie dátových tokov, táto práca sa zameriava predovšetkým na prácu s touto knižnicou.
- **libavdevice** – zabezpečuje funkcionalitu potrebnú pre prácu s rôznymi multimediálnymi zariadeniami.
- **libswscale** – knižnica pre prácu s obrázkami (zmena veľkosti, práca s pixelmi, atď.).
- **libswresample** – knižnica pre prácu s audio streamom (zmena bitovej rýchlosti, konverzia formátu, apod.).

a nástrojmi:

- **ffmpeg** – nástroj pre rýchlu konverziu audio a video súborov alebo ich filtrovanie pomocou filtrov z knižnice libavfilter.
- **ffserver** – streamovací sever pre audio a video.
- **ffplay** – jednoduchý prehrávač využívajúci FFmpeg a SDL knižnice.
- **ffprobe** – nástroj, ktorý dokáže získať informácie o multimediálnom súbore a vypísať ich v zrozumiteľnej podobe.

Ako bolo spomenuté v popise jednotlivých knižníc FFmpegu, táto práca sa bude venovať hlavne práci s knižnicou libavfilter. FFmpeg poskytuje radu filtrov, ktoré umožňujú manipuláciu so vstupnými video dátami ako napríklad orezať zobrazovanú plochu (filter crop), pridávať útvary (filtre drawbox, drawgrid, ...), zmenu počtu snímkov za sekundu (filter framerate) a mnoho ďalších úprav.

Syntax grafu filtrov v FFmpegu

Reťaz filtrov grafov pozostáva z postupnosti filtrov, kde je každý filter napojený na predchádzajúci filter postupnosti. Reťaz filtrov je reprezentovaná ako zoznam filtrov oddelených znakom ",". *Graf filtrov* pozostáva z postupnosti reťazí filtrov, kde sú jednotlivé reťaze oddelené znakom ";".
Filter je reprezentovaný reťazcom v tvare

```
[inLink1]...[inLinkN]filterName=arguments[outLink1]...[outLinkM]
```

filterName je názov triedy filtru, ktorej inštanciou daný filter je. Trieda musí byť registrovaná v programe. Názov triedy filtru môže byť voliteľne nasledovaný reťazcom *=arguments*. *arguments* je reťazec pozostávajúci z parametrov použitých ku inicializácii inštancie filtru. Môže byť v dvoch formách:

- Zoznamom párov *key=value* oddelených znakom ":"

- Zoznamom *value* oddelených znakom ":"

Kde *key* je názov vlastnosti filtru a *value* je hodnota vlastnosti filtru, na ktorú má byť inicializovaná. Ak je hodnota *value* sama o sebe zoznamom hodnôt, sú jednotlivé hodnoty oddelené znakom "|". Zoznam arguments môže byť uzavretý v jednoduchých úvodzovkách ako začiatočný a konečný znak, v tom prípade môžu byť v reťazci použité escape-sekvencie (pomocou znaku "\"). V opačnom prípade je reťazec argumentov ukončený nasledujúcim špeciálnym znakom z množiny znakov "[]=;,".
Názov triedy filtru alebo zoznam argumentov môže byť voliteľne predchádzaný alebo nasledovaný zoznamom značiek určujúcich prepojenie filtrov, ktoré označujú vstupné alebo výstupné piny. Ak sú nájdené dve značky s rovnakým názvom, je medzi filtrami ku ktorým náležia vytvorené spojenie [5].
Aplikáciu týchto pravidiel na príklade ilustruje nasledujúca časť.

Príklad grafu filtrov v ffplay

Vstupný súbor video súbor sa bude označovať ako *inVideo*. Za inVideo sa môže dosadiť ľubovoľný video súbor, napríklad clock.avi. Ako bolo spomenuté v predchádzajúcej časti, nástroj *ffplay* slúži k prehrávaniu videí. Vstupné video (inVideo) sa pomocou ffplay spustí príkazom na príkazovom riadku:

```
ffplay inVideo
```

Zvolený video súbor sa týmto príkazom prehrá v okne vytvorenom pomocou knižnice SDL[3]. Pridaním parametru *–vf* nasledovaného textovým reťazcom sa na vstupný súbor aplikuje graf filtrov alebo vypíše chybové hlásenie, ak niektorý zo zadaných filtrov neexistuje alebo nie je dodržaná syntax grafu filtrov popísaná v 1.7. Filtrom sa môžu pridávať parametre, napríklad filter *crop* (orezanie vstupného videa na zadané rozmery) má štyri voliteľné parametre w (width, šírka výstupného videa), h (height, výška výstupného videa), x (horizontálna pozícia vo vstupnom videu, bod od ktorého sa video orezáva) a y (vertikálna pozícia vo vstupnom videu). Filter crop so všetkými spomenutými parametrami sa potom zapíše napríklad *crop=w=256:h=256:x=0:y=0* alebo jednoduchšie s vynechaním názvov parametrov *crop=256:256:0:0*. Tento filter sa na video aplikuje príkazom:

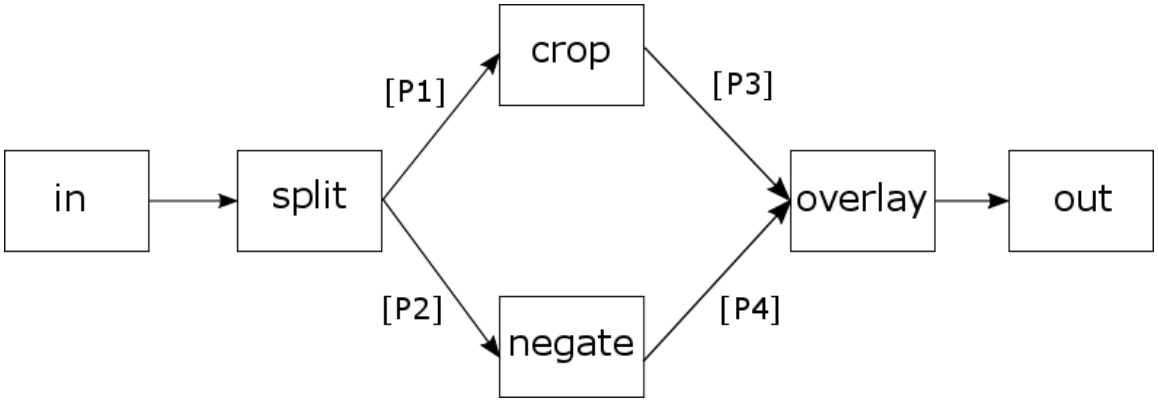
```
ffplay -vf "crop=256:256:0:0" inVideo
```

Pomocou reťazcov filtrov možno pomocou vetvenia zostaviť ľubovoľný graf filtrov. Jednotlivé filtre sa od seba oddeľujú bodkočiarkou (;). V prípade, že má filter viac ako jeden vstup alebo výstup, t.j. má viac vstupných alebo výstupných padov, sa tieto pady pomenujú názvom uvedenom v hranatých zátvorkách. Názvy vstupných padov (v hranatých zátvorkách) sa uvádzajú pred názov filtru a názvy výstupných padov sa uvádzajú za názov filtru. Príkladom reťazenia filtrov s pomenovanými padmi je:

```
ffplay -vf "[in] split [P1][P2]; [P2] negate [P4]; [P1] crop=iw:ih/2:0:ih/2
[P3]; [P4][P3] overlay=0:H/2 [out]" inVideo
```

Použitý graf filtrov sa lepšie znázorní pomocou obrázku 1.6.

[3] https://www.libsdl.org/



Obr. 1.6: Graf filtrov v ffplay

Nie všetky pady musia byť pomenované. Vtedy sú nepomenované pady poprepájané podľa nasledovného pravidla. Prvý nepomenovaný výstupný pad jedného filtru je spojený s prvým nepomenovaným vstupným padom druhého filtru. V takomto prípade sa ako oddeľovač miesto bodkočiarky používa čiarka (,) [1][3].

Kapitola 2

Popis návrhu a implementácie aplikácie

V súčasnosti existuje niekoľko aplikácií, ktoré pracujú s grafmi filtrov rôznych multimediálnych frameworkov. Na začiatku tejto kapitoly sú predstavené niektoré takéto existujúce riešenia, ktoré poslúžili ako zdroj inšpirácie a nápadov. Nasledujúce podkapitoly sa potom venujú výberu programovacieho jazyka, výberu frameworku potrebného pre implementáciu GUI, návrhu spôsobu uloženia a manipulácie s dátami a návrhu vzhľadu GUI a jeho prvkov. Na úvod krátke zhrnutie, aké funkcie má byť aplikácia schopná vykonávať:

- otvorenie a dekódovaie video súborov rôznych formátov
- tvorba a manipulácia s grafom filtrov - pridávanie a odoberanie filtrov
- zmena parametrov jednotlivých filtrov grafu
- vytváranie spojení medzi pinmi filtrov
- zápis výstupu po filtrovaní do súboru
- generovanie grafu prostredníctvom importu reťazca obsahujúceho graf filtrov použiteľného v ffplay (viď. 1.7)
- zobrazenie náhľadu na výsledok aplikácie grafu filtrov na vstupné video
- Exportovanie grafu filtrov vo forme reťazca pre nástroj ffplay

V prípade pridávania filtrov do grafu filtrov a vytvárania spojení medzi filtrami (pinmi) je požadovaná podpora funkcie Drag and Drop (preťahovanie pomocou myši). Výsledná aplikácia sa bude nazývať Filter Graph Editor.

2.1 Existujúce riešenia pracujúce s grafmi filtrov

Ako zdroj inšpirácie pre návrhu aplikácie boli použité dva grafické nástroje pre prácu s grafmi filtrov – GstEditor a GraphEdit. GstEditor je nástroj postavený nad GStreamerom. GraphEdit je dodávaný spoločne s DirectShow, ktorý je zároveň aj jeho základom. Vyskúšanie si práce s týmito dvoma programami napomohlo k získaniu predstavy o tom, ako sa dá realizovať zobrazovanie filtrov ako uzlov grafov, ako znázorňovať piny a spojenia filtrov a v neposlednej rade spôsob akým aplikáciu ovládať pomocou myši.

2.2 Výber programovacieho jazyka a GUI toolkitu

Možými implementačnými programovacími jazykmi sú C alebo C++. Vzhľadom na to, že v prípade tejto práce nie je rýchlosť moc podstatná a tvorba aplikácií s grafickým používateľským rozhraním (GUI) je podstatne jednoduchšia a pohodlnejšia vo frameworku napísanom v jazyku podporujúcom objektovo-orientované programovanie. Preto bude ako implementačný jazyk použitý jazyk C++. Pre jazyk C++ existuje rada GUI toolkitov resp. frameworkov. Za najlepšie možné voľby osobne považujem Qt, Gtkmm a wxWidgets. Všetky tri spomínané frameworky sú multiplatformové (prenositeľné). V tomto prípade som sa rozhodol pre Qt z dôvodu osobných skúseností s týmto toolkitom, ktoré mi v ostatných spomínaných prípadoch chýbajú.

2.2.1 Qt framework

Qt je multiplatformový framework rozširujúci možnosti jazyka C++ pomocou preprocesoru nazývaného Meta-Object Compiler (MOC). Novými vlastnoťami jazyka sa tak stanú napríklad sloty, signály a ďalšie. Pred samotnou kompiláciou Meta-Object Compiler zanalyzuje zdrojové texty napísané v Qt-obohatenom C++ a následne vygeneruje štandardné C++ zdrojové texty, ktoré sa potom dajú skompilovať pomocou štandardných C++ prekladačov ako napríklad GCC, MSVC, Clang, apod [8].

Najdôležitejšou novou pridanou vlastnosťou Qt sú už vyššie spomínané **signály a sloty**, ktoré sa používajú na zabezpečenie komunikácie medzi objektmi. Iné frameworky takúto komunikáciu dosahujú pomocou tzv. callbackov - ukazovateľov na funkcie. Callbacky sú narozdiel od signálov a slotov neintuitívne a môžu trpieť problémami pri zaručení správnosti typov parametrov callbacku.

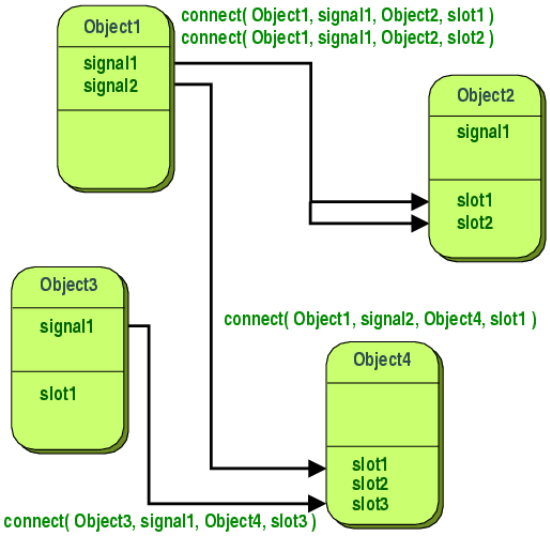
V Qt je signál vyslaný (pomocou makra emit) pri určitej udalosti. Qt widgety majú preddefinovné množstvo signálov, ale môžu byť vytvorením podtried rozšírené o vlastné signály. Slot je metóda, ktorá je volaná ako odpoveď na vyslaný signál. Podobne ako signály aj sloty môžu byť pridávané do vlastných podtried. Systém signálov a systémov je typovo bezpečný - signatúra signálu sa musí zhodovať so signatúrou slotu, ale sloty môžu ignorovať niektoré odoslané parametre, t.j. signatúra slotu môže byť kratšia ako signatúra signálu. Jedine objekty dediace z triedy QObject môžu využívať mechnizmus slotov a signálov. Aby objekt mohol slotom reagovať na signál, musí byť najprv jeho slot s vysielaným signálom spojený. Takéto spojenia je možno vytvárať a rušiť dynamicky za behu programu. Obrázok 2.1 znázorňuje systém signálov a slotov [11].

Widgety sú základné elementy pre tvorbu používateľksých rozhraní v Qt. Môžu zobrazovať dáta, informácie, prijímať používateľské vstupy alebo poskytovať kontajner pre iné widgety. Widget, ktorý nie je vstavaný do rodičovského widgetu sa nazýva okno (window) [10].

Dialógové okná sú sekundárne okná poskytujúce používateľovi možnosti alebo voľby. Dialógové okná môžu byť modáln, v tom prípade musí používateľ zadať požadované informácie pre tým ako môže hlavné okno pokračovať v práci. Naopak nemodálne okná neblokujú prácu hlavného okna.

2.3 Návrh backendu

Táto podkapitola sa bude venovať návrhu backendu. Konkrétne sa bude jednať o riešenie spôsobu tvorby zoznamu filtrov FFmpegu a definovanie objektového modelu grafu filtrov.



Obr. 2.1: Signály a sloty v Qt (Zdroj: [11])

2.3.1 Návrh objektového modelu grafu filtrov

Dôležitou časťou práce je správne navrhnutý objektový model reprezentujúci graf filtrov. Ideálnym spôsobom ako ho realizovať bude definovať vlastné triedy predstavujúce základné elementy grafu filtrov. Týmito triedami sú:

- Pipeline
- Filter
- FilterProperty
- Pin

Trieda *Pipeline* poslúži ako "kontajner"pre filtre a sprostredkuje operácie nad grafom filtrom ako sú pridanie filtru, odobratie filtru alebo aplikácia grafu filtrov na vstupné video. Okrem toho umožní vygenerovať z obsahujúcich filtrov reťazec grafu filtrov použiteľný v nástroji ffplay.

Trieda *Filter* bude reprezentovať FFmpeg filter. V prvom rade bude musieť zaručiť, že filter s daným menom skutočne existuje v FFmpegu. Knižnica FFmpegu libavfilter obsahuje funkciu `avfilter_next`, ktorá iteruje nad všetkými registrovanými filtrami. Opakovaným volaním tejto funkcie sa dá získať kompletný zoznam registrovaných filtrov. Z týchto filtrov potom treba vyčleniť tie, ktoré pracujú s video stopami.

Trieda *FilterProperty* bude predstavovať vlastnoti filtrov. Zoznam vlastností filtrov je možné získať pomocou funkcie `av_opt_next`. Trieda zabezpečí, aby nebolo možné hodnoty vlastností nastaviť na hodnoty neplatné pre typ danej hodnoty. Hodnoty vlastností môžu spadať do kategórií celočíselná hodnota (int, bool, ...), číslo s plávajúcou desatinnou čiarkou (float, double), reťazec alebo racionálne číslo.

Trieda *Pin* bude predstavovať prípojné body filtrov a bude uchovávať informácie o vytvorenom spojení. Filter sám o sebe nebude mať o spojeniach žiadne informácie, ale bude o ne musieť požiadať pin. Spojenie bude vytvoriteľné len medzi pinmi rozdielnych typov (výstupný so vstupným) a názov spojenia sa vytvorí z názvov spájaných pinov a

filtrov vo formáte `Filter1Id_out_OutPinOrder_FilterDestId_in_InPinOrder`, kde FilterSrcId je identifikátor filtru, ktorý obsahuje spájaný výstupný pin číslo OutPinOrder a FilterDestId je identifikátor filtru obsahujúci spájaný vstupný pin a číslom InPinOrder. Napríklad spojením výstupného pinu filtru s identifikátorom filter1 a vstupného pinu filtru s identifikátorom filter2,kde oba filtre majú len jeden pin vznikne spojenie s názvom `filter1_out_0_filter2_in_0`. Za predpokladu, že filter1 je identifikátor filtru triedy scale a filter2 je identifikátor filtru triedy crop, by v reťazcovej reprezentácii grafu filtrov toto spojenie malo tvar:

```
... scale [filter1_out_0_filter2_in_0];[filter1_out_0_filter2_in_0] crop ...
```

2.3.2 Uloženie grafu filtrov

Spôsobov ako uložiť alebo reprezentovať sa dá s trochou fantázie vymyslieť nespočetne mnoho. Avšak najvhodnejším spôsobom ukladanie grafu filtrov bude ukladať ich vo forme reťazca, aký by sa použil pri filtrovaní videa pomocou nástroja *ffplay* (viď. 1.7).

V navrhovanej aplikácii sa bude využívať varianta s povinne uvedenými názvami vstupných a výstupných pinov, aj v prípade ak má filter len jeden vstup alebo jeden výstup. Je to hlavne z dôvodu zjednodušenia logiky vytvárania reťazcov predstavujúcich graf filtrov v aplikácii, kde by bolo zložité zaručiť správnosť prepojenia nepomenovaných filtrov. A rovnako z dôvodu, že ako oddeľovač jednotlivých filtrov sa môže používať len bodkočiarka. Čo umožní jednoduchšie spracovanie uložených grafov filtrov. Grafy filtrov budú uložené v textovej podobe v súboroch s koncovkou *.fge (Filter Graph Editor)*. Súbor bude obsahovať 2 typy záznamov začínajúcich buď slovom *Filter* alebo *Connection*. Záznamy Filter budú definovať uložený filter a ich štruktúra bude:

```
Filter;name;id;x;y;w;h;in;out;params
```

Kde *name* je názov filtru, *id* je unikátny identifikátor filtru v rámci aplikácie, x a y sú súradnice filtru na pracovnej ploche aplikácie, w a h sú výška a šírka, *in* a *out* sú počty vstupných, resp. výstupných pinov a params sú zadané parametre filtru vo formáte *param1=val1,param2=val2,...,paramN=valN*. Parametre filtrov, ktoré nie sú zadané alebo majú implicitnú hodnotu sa ukladať nebudú.

Záznamy Connection budú predstavovať prepojenia medzi filtrami a budú mať formát:

```
Connection;src;srccp;st;dstcp
```

Kde *src* je identifikátor filtru, z ktorého tok dát vychádza, pripojený na výstupný pin, *srccp* je číslo prípojného bodu (viď Návrh frontendu), *dst* je identifikátor filtru, do ktorého tok dát vstupuje, pripojený na vstupný pin a *dstdp* je číslo vstupného bodu filtru.

Import grafu filtrov vo forme reťazca

Aplikácia by mala byť schopná prijať reťazec obsahujúci graf filtrov, ktorý je možné použiť ako parameter nástroja ffplay (viď. 1.7). Z dôvodu zjednodušenia spracovania sa vstupný reťazec najprv predspracuje. Nad reťazcom sa vykonajú nasledujúce operácie:

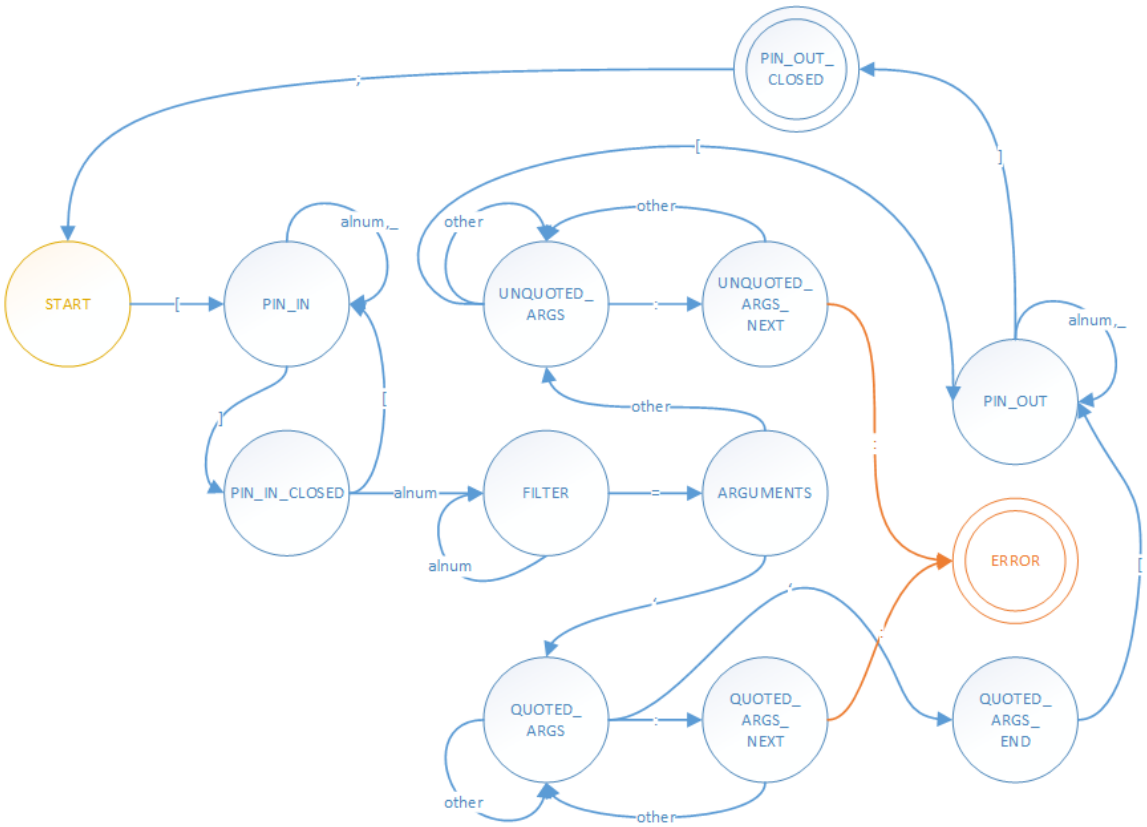
1. Overí sa, že vstupný reťazec obsahuje piny [in] a [out] a to práve jeden z každého, v

opačnom prípade je reťazec neplatný a spracovanie sa ukončí chybou.

2. V reťazci sa nahradia všetky výskyty čiarky (",") unikátnymi značkami vo formáte `[label_x];[label_x]`, kde *label* bude časové razítko a x poradie výskytu čiarky v reťazci začínajúce číslom 0. Použitím časového razítka sa dostatočne zabezpečí unikátnosť značky. Nahradením všetkých čiarok za značky sa docieli to, že si aktuálne spracovaný filter nebude musieť pamätať predchádzajúci filter.

3. Z reťazca sa odstránia všetky biele znaky.

Takto upravený reťazec sa potom spracuje pomocou konečného automatu uvedeného na obrázku 2.2. Na obrázku je z dôvodu prehľadnosti vynechaná väčšina prechodov do stavu ERROR, do ktorého sa prechádza ak nebola splnená ani jedna podmienka. *Alnum* označuje všetky alfanumerické znaky a *other* všetky ostatné možnosti, stavy s podmienkami other nemajú prechod do stavu ERROR. Začiatočným stavom konečného automatu je stav START, jediným koncovým stavom signalizujúcim úspešné pracovanie je stav PIN_OUT_CLOSED.



Obr. 2.2: Konečný automat pre spracovanie grafu filtrov FFmpegu

Stavy konečného automatu:

- START - počiatočný stav alebo začiatok ďalšieho filtru
- PIN_IN - čítanie názvu vstupného pinu filtru
- PIN_IN_CLOSED - úspešne načítaný názov vstupného pinu

- FILTER - čítanie názvu filtru
- ARGUMENTS - čítanie zoznamu vlastností filtru
- UNQUOTED_ARGS - čítanie vlastností filtru, ktoré nie sú ohraničené úvodzovkami
- UNQUOTED_ARG_NEXT - čítanie ďalšej vlastnosti filtru, vlastnosti ako celok nie sú ohraničené úvodzovkami
- QUOTED_ARGS - čítanie vlastností filtru, ktoré sú ohraničené úvodzovkami
- QUOTED_ARG_NEXT - čítanie ďalšej vlastnosti filtru, vlastnosti ako celok sú ohraničené úvodzovkami
- QUOTED_ARGS_END - úspešne načítané vlastnosti ohraničené úvodzovkami
- PIN_OUT - čítanie názvu vstupného pinu filtru
- PIN_OUT_CLOSED - úspešne načítaný názov vstupného pinu, ak bol načítaný posledný znak reťazca grafu filtrov, tak signalizuje úspešné spracovanie
- ERROR - načítaný neočakávaný znak

Úspešným spracovaním sa získa zoznam filtrov, voliteľne ich vlastnosti, a značky určujúce, s ktorými filtrami sú prepojené. Potom pomocou značiek sa presne určí ktorý filter je s ktorým prepojený a zoznam sa pretransformuje do formátu používaného na ukladanie grafov filtrov popísanom v časti 2.3.2.

2.4 Návrh frontendu

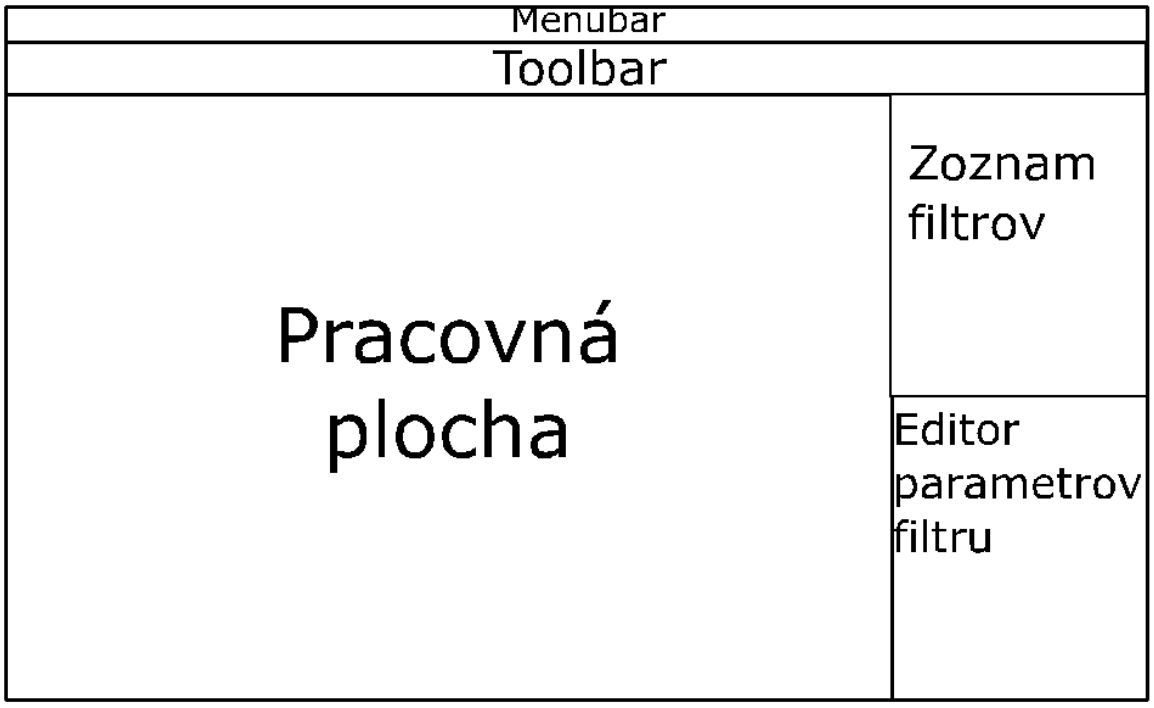
Táto časť sa bude venovať návrhu grafick0ho používateľského rozhrania aplikácie. Načrtáva vzhľad hlavného okna, rozloženie prvkov a ich charakteristiku a na záver popisuje spôsob zobrazovania filtrov ako grafických prvkov.

Hlavné okno

Hlavné okno predstavuje základ celého používateľského rozhrania. Na obrázku 2.3 je načrtnutý vzhľad hlavného okna. Vzhľadom na to, že vlastnosti a názvy filtrov FFmpegu sú v anglickom jazyku, budú aj všetky textové prvky aplikácie v angličtine.

Hlavnými prvkami okna sú:

- **Menubar** – Kontajner pre rozbaľovacie ponuky známe z väčšiny aplikácií s grafickým používateľským rozhraním. Bude obsahovať dve ponuky - File (Súbor) a Filter. Akcie z ponuky File budú otvoriť uložený graf filtrov, uložiť aktuálny graf filtrov, nový graf filtrov (zrušiť aktuálny graf filtrov), aplikovať graf filtrov, zobraziť náhľad na výsledok, import a export pipeline, zobraziť informácie o aplikácii a ukončiť program. Akcie z ponuky Filter budú pracovať nad momentálne vybraným filtrom a budú to akcie pripojiť, odpojiť, pridať a odobrať vstupný alebo výstupný pin a zmazať filter. Dostupnosť týchto akcií bude závisieť od momentálneho stavu filtru.
- **Toolbar** – Kontajner pre tlačidlá poskytujúce najdôležitejšie funkcie aplikácie – bude obsahovať rovnaké akcie ako ponuka File z menubaru s výnimkou zobrazenia informácií o aplikácii.



Obr. 2.3: Návrh rozloženia prvkov v hlavnom okne

- **Pracovná plocha** – Priestor pre prácu s grafom filtrov. Filtre sa na pracovnú plochu budú vkladať prostredníctvom Drag and Drop zo zoznamu filtrov.
- **Zoznam filtrov** – Zoznam všetkých video filtrov FFmpegu.
- **Editor parametrov filtru** – Zobrazí parametre aktuálne vybraného filtru na pracovnej ploche a umožní zmeny týchto parametrov. Zobrazenie filtrov sa bude realizovať ako tabuľka s dvoma stĺpcami – názov parametru a hodnota parametru – a počtom riadkov odpovedajúcemu počtu parametrov filtru plus riadky pre identifikátor a názov filtru, výšku, šírku a súradnice grafického objektu reprezentujúceho filter na pracovnej ploche.

Grafická reprezentácia filtru

Filtre na pracovnej ploche budú reprezentované ako štvoruholník obsahujúci textové pole s názvom a identifikátorom filtru so štyrma prípojnými označenými ako top, bottom, left right, viď obrázok 2.4. Grafické objekty filtrov budú voľne premiestňovateľné po pracovnej ploche s nastaviteľnými rozmermi v rámci určitého intervalu hodnôt a možnosťou vytvárať a rušiť spojenia medzi sebou.

2.5 Štruktúra projektu

Projekt je rozdelený na štyri časti, z ktorých tri dokážu pracovať samostatne bez prítomnosti zvyšných častí. Tento postup som zvolil predovšetkým z dôvodu zjednodušenia testovania jednotlivých častí a oddelenie častí využívajúce FFmpeg od častí využívajúcich Qt.

Obr. 2.4: Návrh grafickej reprezentácie filtru na pracovnej ploche

Spomínanými podprojektmi sú:

- Pipeline
- Workspace
- PipelineParser
- FilterGraphEditor

Prvé tri vymenované projekty sú knižnice, ktoré môžu byť v budúcnosti použité v iných projektoch. FilterGraphEditor zlučuje funkcionalitu predošlých projektov a dodatočne pridáva novú vlastnú funkcionalitu. Nasledujúce časti sa podrobnejšie venujú všetkým vymenovaným projektom.

2.6 Projekt Pipeline

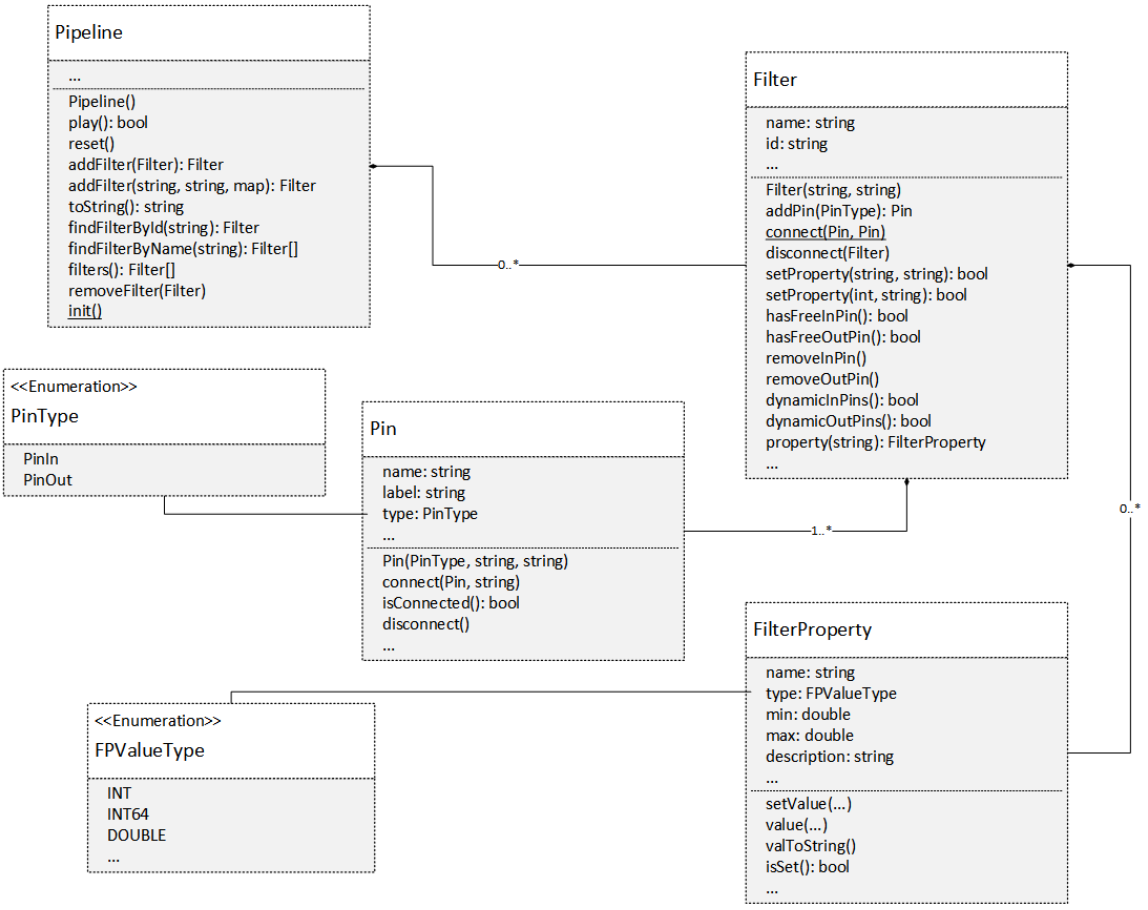
Projekt Pipeline definuje triedy spomínané v kapitole 2.3.1 a doplňuje dve pomocné triedy. V projekte sa používajú len štandardné knižnice C++ a knižnice FFmpegu. Kompletný zoznam definovaných tried je:

- Pipeline
- Filter
- FilterProperty
- Pin
- PipelineError
- Utils

Na obrázku 2.5 je zjednodušený diagram tried projektu Pipeline. Z dôvodu veľkého množstva metód a atribútov niektorých tried sú vymenované len tie najdôležitejšie.

2.6.1 Trieda Pipeline

Trieda **Pipeline** poskytuje prácu s grafom filtrov. Umožňuje pridávať a odoberať filtre do resp. z grafu a potom pridané filtre aplikuje na vstupný súbor. Okrem toho poskytuje rozhranie pre prístup a vyhľadávanie pridaných filtrov. Vyhľadávanie filtrov môže byť podľa názvy triedy filtru, v tom prípade sa vracia zoznam všetkých filtrov danej triedy, alebo na



Obr. 2.5: Zejdnodušený diagram tried projektu Pipeline

základe identifikátoru filtru, kedy sa vráti práve jeden alebo žiaden filter. Pri pridávaní filtru zaručuje unikátnosť identifikátoru filtru.

Pri aplikácii grafu filtrov na vstupné video sa najprv z pridaných a vzájomne poprepájaných filtrov vytvorí reťazec grafu filtrov (viď. 1.7). Zo získaného reťazca sa zistí, či boli pridané vstupy a výstupy, tak že sa skontroluje existencia pinov [in] a [out]. Ak neboli nájdené oba piny, fitrovanie končí chybovou hláškou *Invalid pipeline*. Ďalšiu kontrolu validity aplikovaného reťazca grafu filtrov vykonáva FFmpeg pomocou funkcie `avfilter_graph_parse_ptr`.

Zrušením inštancie triedy sa zrušia aj všetky inštancie tried Filter, ktoré rušená Pipeline obsahuje.

2.6.2 Trieda Filter

Trieda **Filter** implementuje rozhranie pre manipuláciu s filtrami. Obsahuje metódy umožňujúce prístup k atribútom filtru - názov triedy filtru, identifikátor a vlastnosti filtru. Ďalej metódy vytvárajúce a rušiace spojenia medzi pinmy filtrov, nastavenie vlastnoti filtru na zadanú hodnotu a statickú metódu, pomocou ktorej sa získa zoznam video filtrov registrovaných v FFmpegu. Nakoniec ešte implementuje metódy, pomocou ktorých je možné zistiť počet, dostupnosť a možnosť pridávania a odoberania pinov filtru.
Zoznam filtrov sa získa pomocou funkcie knižnice libavfilter `avfilter_next`, ktorou iteráciou nad všetkými filtrami FFmpegu (video, audio a iné filtre) sa postupne pre každý filter

analyzuje typ dátového toku, s ktorým pracuje. Filtre nepracujúce s videom sú preskakované. Do zoznamu sú pridané dva nové filtre *in* a *out*, ktoré predstavujú vstup, resp. výstup grafu filtrov (piny [in] a [out]). Oba filtre sa musia v grafe filtrov vyskytovať práve jeden krát. Ich vlastnosťami sú cesta k vstupnému resp. výstupnému súboru.

Pri vytváraní inštancie triedy Filter sa pred inicializáciou vlastností filtru zistia dodatočné informácie. Podľa názvy triedy filtru sa pomocou funkcie knižnice libavfilter `avfilter_get_by_name` vyhľadá požadovaný filter. Ak daný filter existuje, zistia sa informácie o počte vstupných a výstupných pinov, kde sa jedná buď o konkrétne číslo (vtedy sa u filtru vytvorí rovnaký počet pinov daného typu) alebo má filter dynamický počet pinov určitého typu a nastaví sa príslušný príznak (vtedy sa inicializuje jeden pin daného typu a ďalšie je možné pridať neskôr). Na záver sa načíta zoznam vlastností filtru a ich hodnoty sa nastavia na implicitné hodnoty. Treba brať v úvahu, že niektoré vlastnosti môžu mať aj aliasy, preto aby sa zamedzilo viac násobnému výskytu určitej vlastnosti je treba aliasy detekovať. Detekcia sa vykonáva porovnaním popisov vlastností, ktoré sú pre všetky aliasy rovnaké a použije sa prvá nájdená z týchto vlastností.

Zrušením inštancie triedy sa zrušia aj všetky piny, ktoré rušená inštancia obsahuje a zrušia sa aj spojenie náležiace týmto pinom.

2.6.3 Trieda FilterProperty

Trieda **FilterProperty** predstavuje určitú vlastnosť filtru. Uchováva informácie o danej vlastnosti a poskytuje metódy pre nastavenie hodnoty. Vlastnosť video filtru môže mať práve jeden typ (*FPValueType*), každý typ spadá do jednej zo štyroch kategórií (uvedenej v zátvorkách):

- Flags (Int)
- Int (Int)
- Int64 (Int)
- Double (Double)
- Float (Double)
- String (String)
- Rational (Rational)
- Binary (String)
- Image size (String)
- Pixel format (Int)
- Sample format (Int)
- Video rate (String)
- Duration (Int)
- Color (String)
- Channel layout (Int)

- Bool (Int)

Vlastnoti kategórie Int používajú ako dátový typ C++ typ long long, kategórie double a string majú ako dátový typ double a std::string a typ Rational je reprezentovaný dvojicou čísel typu integer. Číselné typy majú spravidla určitý interval platných hodnôt daných intervalom $\langle min, max \rangle$. Pri nastavovaní hodnoty zaručuje, že hodnota nebude nastavená na neplatnú hodnotu alebo v prípade číselných typov na hodnotu mimo povolený interval hodnôt.

2.6.4 Triedy Pin, PipelineError a Utils

Piny predstavujú prípojné body filtrov a trieda **Pin** implementuje ich realizáciu v projekte. Uchováva informáciu o vytvorenom spojení s pinom opačného typu, informáciu o type pinu a názve spojenia (ak existuje).V triede **PipelineError** sú definované chybové kódy a im náležiace stavové hlásenia. Nakoniec trieda **Utils** implementuje dve pomocné metódy - metódu pre odstránenie prebytočných bielych znakov z reťazca a metódy pre spracovanie racionálnych čísel (viď. 2.6.3) na dvojicu celých čísel.

2.6.5 Graf filtrov v projekte Pipeline

Pomocou vyššie popísaných tried je zostaviteľný ľubovoľný graf filtrov. Nasledujúci príklad demonštruje graf filtrov obsahujúci 2 FFfpeg filtre - hflip a scale, reťazec status obsahuje stavovú informáciu - chyba alebo úspech:

```
Pipeline::init();
Pipeline *pipeline = new Pipeline();

Filter *fin    = pipeline->addFilter("fin" ,"in",
    {{"file", "clock.avi"}});
Filter *fout   = pipeline->addFilter("fout", "out",
    {{"file", "test.avi"}, {"format", "avi"}});
Filter *fscale = pipeline->addFilter("fscale", "scale",
    {{"width", "iw/2"}, {"height", "ih/2"}});
Filter *fhflip = pipeline->addFilter("fhflip", "hflip");

Pin *in_out    = fin->firstFreeOutPin();
Pin *scale_in  = fscale->firstFreeInPin();
Pin *scale_out = fscale->firstFreeOutPin();
Pin *hflip_in  = fhflip->firstFreeInPin();
Pin *hflip_out = fhflip->firstFreeOutPin();
Pin *out_in    = fout->firstFreeInPin();

Filter::connect(in_out, scale_in);
Filter::connect(scale_out, hflip_in);
Filter::connect(hflip_out, out_in);

std::cout << pipeline->toString() << std::endl;

std::string status;
pipeline->play(status);
```

```
std::cout << status << std::endl;
```

2.7 Projekt Workspace

Dôvodom vzniku tohto projektu bolo vytvoriť knižnicu umožňujúcu vytvárať widgety, ktoré sú voľne presúvateľné po pracovnej ploche. Väčšina tu implementovaných tried je určená k subclassingu a preto má mnoho metód prázdne telo definície, aj napriek tomu možno pomocou týchto tried vytvoriť nehodnotený orientovaný graf. Projekt využíva Qt a implementuje tieto triedy:

- Workspace
- WorkspaceItem
- WorkspaceItemConnection
- WorkspaceItemLabel
- WorkspaceItemSelector

2.7.1 Trieda Workspace

Inštancia triedy **Workspace** predstavuje pracovnú plochu, na ktorú možno pridávať objekty typu WorkspaceItem. Definuje veľké množstvo Qt signálov a slotov a spracovanie udalostí vyvolaných pohybom myši alebo kliknutím niektorého tlačítka myši. Uchováva informácie o všetkých objektoch WorkspaceItem a ich spojeniach a ukazovatele na aktuálne vybraný objekt, ktorého zmena vyšle signál informujúci o zmene, na objekt, ktorý zahajuje spojenie s iným objektom a objekt, ktorý sa chce odpojiť od niektorého z pripojených objektov. Ďalej informuje prostredníctvom signálov o zmene rozmerov a pozície aktuálne vybratého objektu a stará sa o vykreslenie šípok zobrazujúcich smer prepojení medzi jednotlivými objektami. Pridaním objektu WorkspaceItem sa vytvoria spojenia medzi signálmi a slotmi pre vzájomnú komunikáciu.

2.7.2 Trieda WorkspaceItem

Trieda **WorkspaceItem** definuje widgety, ktoré reprezentujú uzly grafu na pracovnnej ploche. Každá inštancia je unikátne identifikovateľná pomocou identifikátoru. Identifikátor je tvorený názvom objektu a číslom symbolizujúcim poradie pridaného objektu na pracovnú plochu. Poradie sa odoberaním (rušením) objektov neznižuje. Objekty WorkspaceItem sú voľne presúvateľné po pracovnej ploche, možno im meniť rozmery na hodnoty v intervale $\langle 50, 300 \rangle$. Možno ich vzájomne prepájať prostredníctvom prípojých bodov, tu nazývaných *Connection point*, ktorých má každý objekt štyri tak ako sú načrtnuté na obrázku 2.4.

Prípojné body sú identifikované hodnotami výčtového typu `WIConnectPoint`:

- CP_TOP - horný prípojný bod
- CP_LEFT - ľavý prípojný bod
- CP_BOTTOM - spodný prípojný bod

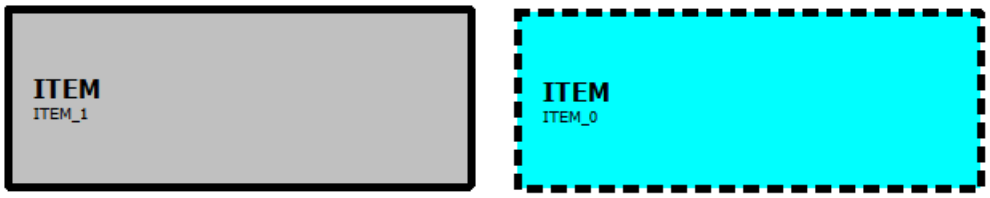
- CP_RIGHT - pravý prípojný bod

Použitím prípojných bodov nie je potrebné uchovávať presnú súradnicu miesta, v ktorom začína alebo končí spojenie, ktorá by sa musela zložito prepočítavať pri zmene rozmerov alebo presúvaní objektu. Takto si stačí uchovávať informáciu o tom, v ktorom zo štyroch bodov spojenie začína alebo končí. Pri zmene pozície alebo rozmerov sa len prepočíta súradnica stredu náležiacej strany.

Objekt sa implicitne nachádza v pohybovom stave (`MODE_MOVE`), dvojklikom na objekt sa zmení stav objektu na stav zmeny rozmerov (`MODE_RESIZE`), ktorý je signalizovaný zmenou okrajov a je v ňom možné meniť rozmery. Prechod do pohybového stavu sa vykoná opätovným dvojklikom na objekt alebo stratou sústredenia (focus). Obrázok 2.6 ilustruje tieto dva stavy. Objekt naľavo sa nachádza v pohybovom stave a objekt napravo sa nachádza v stave zmeny rozmerov.

WorkspaceItem definuje metódy pre vytváranie a rušenie spojení medzi sebou. Spojenie sa vždy vedie od zdroja ku cieľu. Spôsobu vytvárania a rušení spojení sa budem ďalej venovať v časti 2.9.3.

Aktuálne vybraný objekt sa dá identifikovať pomocou azúrového sfarbenia, objekty, od ktorých sa môže zvolený objekt odpojiť sú sfarbené na fialovo, ostatné objekty sú sfarbené na šedo. Na obrázku 2.6 je objekt vpravo aktuálne vybraným objektom.



Obr. 2.6: Stavy objektov WorkspaceItem, ITEM_0 je momentálne vybraný objekt

2.7.3 Triedy WorkspaceSelector, WorkspaceItemLabel a WorkspaceItemConnection

Trieda **WorkspaceSelector** predstavuje widget, ktorý poskytuje zoznam dostupných typov WorkspaceItem a editor vlastností objektov typu WorkspaceItem. Poskytuje sloty reagujúce na signály vyslané z Workspace, ale väčšina metód má prázdne telo a očakáva sa, že táto trieda sa použije ako rodičovská trieda inej triedy. Trieda **WorkspaceItemLabel** predstavuje popisok zobrazený na objekte WorkspaceItem obsahujúci názov objektu a identifikátor objektu. WorkspaceItemConnection predstavuje spojenie medzi dvoma inštanciami triedy WorkspaceItem, v ktorom ako zdroj (source) vystupuje inštancia, z ktorej bolo spojenie inicializované a ako cieľ (destination) inštancia, do ktorej bolo spojenie vedené.

2.8 Projekt PipelineParser

V projekte PipelineParser sú realizované návrhy popísané v 2.3.2. Projekt využíva Qt. Obsahuje dve triedy:

- PipelineParser
- ParsedFilter

V triede **PipelineParser** je implementované spracovanie vstupného reťazca a konečný automat 2.2. Filtre a informácie načítané zo vstupného reťazca sa ukladajú v inštanciách triedy **ParsedFilter** obsahujúce názov filtru, poradie výskytu v reťazci, vstupné a výstupné piny a vlastnosti. Pri transformácii získaných filtrov na formát uložených filtrov (viď. 2.3.2) sa filtrom prideľujú súradnice podľa poradia výskytu: $y = 150$, $x = poradie * 150$. Výška je vždy 80 a šírka 100.

2.9 Projekt FilterGraphEditor

V projekte FilterGraphEditor je zlúčená funkcionalita naimplementovaná v predchádzajúcich projektoch. Väčšina tried z projektu Workspace je tu použitá ako rodičovská trieda niektorej z tu definovaných tried. Tieto nové triedy sú potom ďalej rozšírené o atribúty, ktoré sú inštanciami tried z projektu Pipeline alebo Pipeline Parser. Kompletný zoznam tried definovaných v tomto projekte je:

- FilterGraphEditor
- EditorWorkspace
- FilterWidget
- FilterSelector
- ReponsivePipeline
- FilteringDialog
- FilteringThread
- ImportPipelineDialog
- PreviewDialog
- AboutDialog

2.9.1 Trieda FilterGraphEditor

Trieda **FilterGraphEditor** predstavuje hlavné okno aplikácie a tým pádom sa stará o rozloženie widgetov ako je ilustrované na obrázku 2.3. Definuje sloty reagujúce na zmenu stavu, ktorej výskyt je potom signalizovaný na StatusBare náležiacim hlásením. Definuje akcie - v kontexte Qt pod pojmom akcia možno rozumieť inštanciu triedy QAction, t.j. elementy rozbaľovacích ponúk (menu) alebo toolbaru. Akciami sú:

- **Open** - otvoriť súbor s uloženým grafom filtrov
- **Save** - uložiť aktuálny graf filtrov do súboru
- **New** - zrušiť aktuálny graf filtrov a začať odznova
- **Import** - importovať pipeline

- **Export** - exportovať pipeline do schránky
- **Filter** - aplikovať aktuálny graf filtrov
- **Preview** - zobraziť náhľad na výsledok filtrovania
- **About** - zobraziť informácie o aplikácii
- **Exit** - ukončiť aplikáciu

2.9.2 Trieda EditorWorkspace

Trieda **EditorWorkspace** je potomok triedy Workspace rozšírenej o atribút inštancie triedy ResponsivePipeline a dialógové okná. Stará sa o ukladanie, načítanie a rušenie grafov filtrov, ktoré sú na nej realizované pomocou vzájomne prepojených inštancií triedy FilterWidget. Ďalšou dôležitou úlohou tejto triedy je vykonanie príslušných operácií ako reakcie na akcie hlavného okna, ku ktorým pridáva niekoľko vlastných akcií a reakcií na ne, nové akcie pracujú s aktuálne vybraným filtrom a sú to:

- **Connect** - zahájiť spojenie s iným filtrom
- **Disconnect** - odpojiť filter od niektorého z pripojených filtrov
- **Delete** - zmazať filter a zrušiť všetky jeho spojenia
- **Add In Pin** - pridať vstupný pin, táto akcia je dostupná len v prípade, že má filter dynamický počet vstupných pinov
- **Add Out Pin** - pridať výstupný pin, táto akcia je dostupná len v prípade, že má filter dynamický počet výstupných pinov
- **Remove In Pin** - odobrať vstupný pin, táto akcia je dostupná len v prípade, že má filter dynamický počet vstupných pinov a aspoň jeden z nich je voľný, nemožno odobrať piny s aktívnym pripojením
- **Remove Out Pin** - odobrať výstupný pin, táto akcia je dostupná len v prípade, že má filter dynamický počet výstupných pinov a aspoň jeden z nich je voľný, nemožno odobrať piny s aktívnym pripojením

Nasledujúce časti popisujú postup pri vykonávaní dôležitých akcií. V nich bude inštancia triedy FilterWidget označovaná ako filter a ako pracovná plocha bude označená inštancia triedy EditorWorkspace.

Connect - Vytváranie spojení medzi filtrami

Vytváranie spojení medzi filtrami je základnou akciou, pomocou ktorej sa vytvára graf filtrov. Vytvorenie spojenia obnáša vykonanie niekoľkých krokov nasledujúcich po použití akcie *Connect* na vybraný filter:

1. Filter si nastaví príznak o tom, že inicializuje spojenie a vyšle signál so žiadosťou pracovnej ploche a zobrazí svoje prípojné body. Pracovná plocha vyzve všetky ostatné filtre, aby schovali svoje prípojné body a nastavili sa do role cieľa. Ak filter nemá voľný výstupný pin, tak musí odvolať svoju žiadosť, toto je popísané v ďalšej časti.

2. Kliknutím na jeden z aktívnych prípojných bodov filtru si uloží informáciu o kliknutom bode a vyšle signál pracovnej ploche, že sa chce pripojiť práve k tomu bodu.

3. Pracovná plocha vyzve všetky filtre s voľným vstupným pinov aby zobrazili svoje prípojné body a nastaví ukazovateľ na žiadateľa. Prípadne sa môže zrušiť predchádzajúca žiadosť o spojenie a/alebo odpojenie.

4. Kliknutím na niektorý z novo zobrazených prípojných bodov si tento filter uloží informáciu o kliknutom bode a vyšle signál pracovnej ploche.

5. Vytvorí sa spojenie medzi filtrami (obsadia sa piny, aktualizuje tooltip) a pracovná plocha si poznamená informaciu o novom spojení - ktoré filtre a cez ktoré prípojné body (viď. WorkspaceItemConnection - 2.7.3).

6. Pracovná plocha nastaví ukazovateľ na žiadateľa na NULL, vyzve filtre aby schovali prípojné body a vykreslí spojenie.

Connect - Odvolanie žiadosti o spojenie

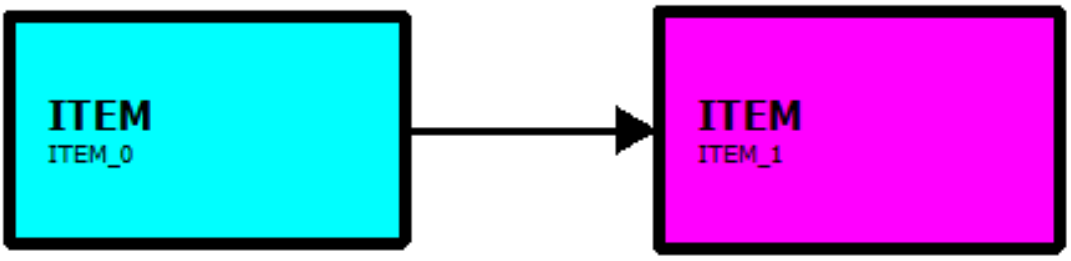
K odvolaniu žiadosti o pripojenie dochádza v prípade ak nemá žiadajúci filter voľný výstupný pin, opätovnej žiadosti o spojenie (*Connect*) alebo ak počas žiadosti jedného filtru požiadal o spojenie iný filter:

1. Filter zruší príznak o tom, že sa chce pripojiť a vyšle signál so žiadosťou o zrušenie žiadosti o spojenie.

2. Pracovná plocha nastaví ukazovateľ na žiadateľa na NULL a vyzve všetky ostatné filtre aby schovali svoje prípojné body, a to aj v prípade, že ich ešte nezobrazili.

Disconnect - Rušenie spojení medzi filtrami

O zrušenie spojenia môže žiadať filter, ktorý má aspoň jedno aktívne spojenie s iným filtrom. Žiadosť o odpojenie začína po použití akcie *Disconnect*. Kroky, ktoré sa počas tohto procesu vykonajú sú takmer identické s krokmi ako v prípade vytvárania spojení. S tým rozdielom, že sa nezobrazujú prípojné body žiadateľa ani zvyšných filtrov, ale odpojiteľné filtre zmenia svoju farbu na fialovú (obr. 2.7). Podobne aj odvolanie žiadosti o odpojenie je podobná odvolaniu žiadosti o pripojenie.



Obr. 2.7: Rušenie spojenia medzi WorkspaceItem

Save a Open - Uloženie a načítanie grafu filtrov

Ukladanie grafu filtrov využíva postup navrhnutý v 2.3.2. Najprv sa pre každý filter pracovnej plochy vytvorí reťazec začínajúci záznamom Filter a potom sa pre každé zaznamenané spojenie medzi filtrami vytvoria reťazce začínajúce záznamom Connection. Reťazce sa priebežne zapisujú do zvoleného súboru.
V prípade načítania je situácia trochu komplikovanejšia. Najprv sa celý dokument riadok po riadku načíta do zoznamu reťazcov (QStringList), kde sa potom jednotlivé riadky analyzujú:

1. Reťazec sa rozdelí na časti, kde ako separátor je znak ";".
2. Porovná sa, či je prvé slovo Filter alebo Connection, inak chyba načítania.
3. Na základe výsledku predošlého kroku sa porovná počet častí pôvodného reťazca. Pre záznamy Filter musí byť počet 10, pre záznamy Connection musí byť počet 5, inak chyba načítania. Ak je záznam typu Filter s odpovedajúcim počtom častí, pokračuje sa bodom 4, inak bodom 8.
4. Reťazce obsahujúce x a y súradnicu, výšku, šírku a počty pinov sa prevedú na celé čísla (integer), ak dôjde pri prevode k chybe, tak chyba načítania.
5. Z identifikátoru sa získa poradové číslo, ktoré sa porovná s počítadlom pridaných widgetov na precovnej ploche, ak je číslo počítadla menšie, priradí sa mu číslo z identifikátoru + 1. Ak pri prevode čísla identifikátoru dôjde k chybe alebo číslo nebolo vôbec nájdené, tak chyba načítania.
6. Zkonštruuje sa nový objekt FilterWidget a inicializujú sa jeho piny. AK je počet načítaných pinov väčší ako počet pinov vytvoreného filtru a zároveň filter nemá dynamický počet pinov daného typu, tak chba načítania.
7. Inicializujú sa vlastnosti filtru, ak nejaké boli načítané. Vlastnosti môžu byť natavené podľa názvu alebo podľa poradia výskytu, ak názvy nie sú uvedené. Ak pri inicializácii vlastností dôjde k chybe, tak chyba načítania.
8. Pre každý záznam typu Connection sa na základe identifikátorov vyhľadajú vytvorené filtre - source a destination, ak aspoň jeden z hľadaných filtrov neexistuje, chyba načítania.
9. Obom filtrom sa nastaví aktívny prípojný bod podľa načítaného čísla.
10. Vytvorí sa spojenie medzi filtrami.

Ak pri načítaní dôjde k chybe, tak sa zavolá akcia New - zmažú sa všetky filtre vytvorené v priebehu načítania.

2.9.3 Trieda FilterWidget

Trieda **FilterWidget** je jednou z najdôležitejších tried projektu. Je potomkom triedy WorkspaceItem z projektu Workspace doplnenej o atribút filter - objekt triedy Filter z projektu Pipeline.

Vytvorenie nového FilterWidgetu

Nový FilterWidget sa vytvorí pretiahnutím jedného z filtrov dostupného z widgetu FilterSelector. Novo pridaný filter má súradnicu [0, 0] v mieste, na ktoré ukazoval kurzor v momente uvoľnenia ľavého tlačítka myši. Rozmery sa prispôsobia veľkosti obsahu - výšku má každý filter rovnakú, ale šírka závisí od dĺžky textu. Novo vytvorený filter sa stáva aktívnym filtrom. Filtru je pridelený identifikátor pozostávajúci znázvu triey filtru a poradia pridaného prvku na pracovnú plochu.

Rušenie filtru

Filter sa zruší spustením akcie *Delete*. Najprv sa zistí, či aktívny filter nemá žiadosť o zahájenie alebo zrušenie spojenia a podľa toho je požiadaný o zrušenie žiadosti. Následne sú zrušené všetky spojenia rušeného filtru a je zrušený samotný filter. Pri zavolaní akcie *New* sa postupne rušia všetky filtre v poradí, akom sú registrované v zozname filtrov pracovnej plochy.

2.9.4 Trieda FilterSelector

Trieda FilterSelector je potomkom triedy WorkspaceItemSelector, ktorá značne rozširuje funkcionalitu svojho predka. V prvom rade definuje spôsob, akým sú do zoznamu filtrov pridávané záznamy a ďalej spôsob aktualizácie tabuľky editoru vlastností filtrov.
Prvky z ponuky filtrov je možné na pracovnú plochu prenášať pomocou myši - Drag and Drop. Na prenos dát z QListView Qt používa vlastný MIME typ[1], ktorý je prijatý pracovnou plochou a následne je z neho dekódovaný názov prenášaného filtru.
Editor vlastností v tabuľke zobrazuje vlastnosti momentálne vybraného filtru pracovnej plochy. Reaguje na signál pracovnej plochy indikujúci zmenu výberu, rozmerov alebo pozície aktívneho filtru a na základe toho aktualizuje svoj obsah. Pri zmene niektorej vlastnosti vysiela signál obsahujúci informáciu o zmenenej vlastnosti, novej hodnote a typy vlastnosti. AK je hodnota menená na neplatnú hodnotu (nesprávny typ, mimo rozsahu, atď.) nedochádza k zmene a pracuje sa s pôvodnou hodnotou.

2.9.5 Triedy ResponsivePipeline, FilteringThread, FilteringDialog a PreviewDialog

Tieto štyri triedy sú si vzájomne veľmi blízke. **ResponsivePipeline** je trieda Pipeline rozšírená o možnosť používateľského zásahu behom spracovania a navrch pridáva možnosť prevodu filtrovaného snímku (YUV420) na QImage (RGB888) k zobrazenie v PreviewDialog. **FilteringThread** je vlákno, v ktorom sa vykonáva filtrovanie a zápis do výstupného súboru alebo prevod snímku na obrázok.
Modálne dialógy **FilteringDialog** alebo **PreviewDialog** sa zobrazia po spustení akcie *Filter* resp. *Preview*. Na základe spustenej akcie sa nastaví mód ResponsivePipeline (filter, preview). Po zobrazení dialóg vyšle signál, na ktorý pracovná plocha reaguje spustením vlákna FilteringThread. FilteringDialog zobrazuje stav zápisu výsledku filtrovania do súboru a PreviewDialog zobrazuje výstupné video alebo nič ak je pipeline chybná. Treba brať v úvahu, že spracovanie snímkov môže chvíľú trvať a tým pádom môže byť zobrazovanie snímkov v PreviewDialog pomalé.

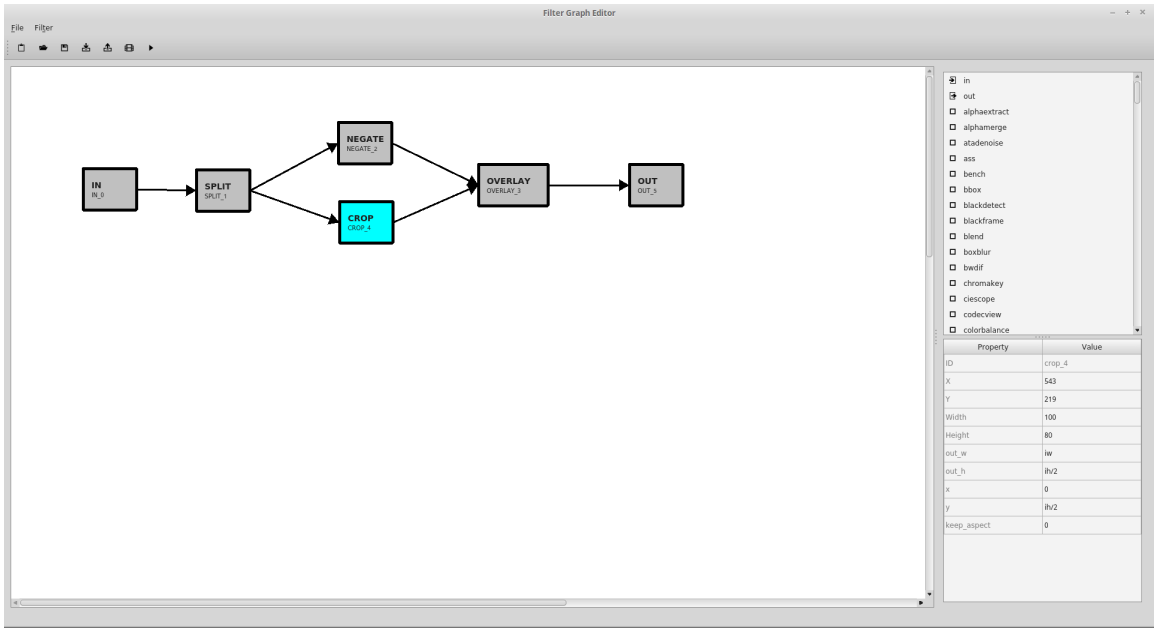
[1] application/x-qabstractitemmodeldatalist

2.9.6 Triedy ImportPipelineDialog a AboutDialog

Tieto triedy implementujú triviálne modálne dialógové okná. **ImportPipelineDialog** je dialóg s textovým poľom, do ktorého sa zadáva reťazec grafu filtrov syntaxou popísanej v 1.7. Vstupný reťazec sa pretransformuje do formátu uloženého grafu filtrov aplikácie a načíta sa ako keby to bol uložený súbor, platí rovnaký postup ako v 2.9.2. **AboutDialog** zobrazí okno s popisom aplikácie, menom autora a rok.

2.10 Aplikácia Filter Graph Editor

Výledkom vyššie popísanej implementácie je aplikácia Filter Graph Editor. Aplikácia umožňuje zostaviť graf filtrov pre FFmpeg, upravovať vlastnosti filtrov, zapísať výstup filtrovania do súboru alebo ho zobraziť v náhľade. Ďalej umožňuje načítať a ukladať graf filtrov alebo ho importovať a exportovať. Na obrázku 2.8 je výsledná aplikácia s jednoduchým grafom filtrov. V aplikácii využívam ikony dostupné na webe http://modernuiicons.com/ šírené pod licenciou Creative Commons Attribution-NoDerivs 3.0 Unported[2]. Obmedzením aplikácie je, že dokáže pracovať len s jedným vstupným a jedným výstupným súborom.

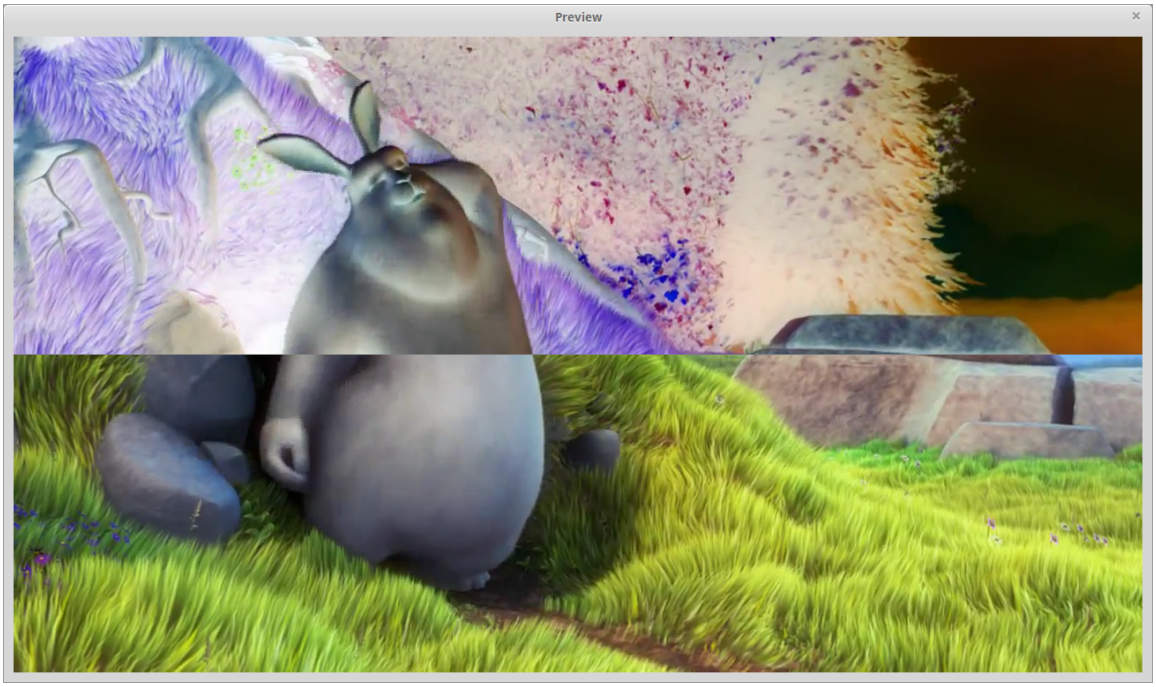


Obr. 2.8: Filter Graph Editor

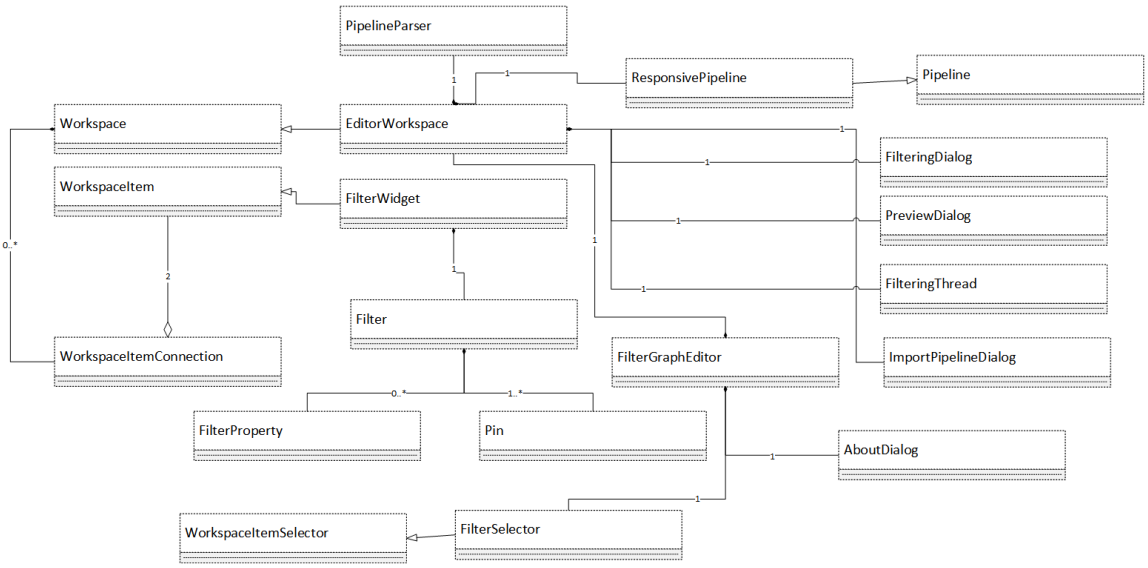
Aplikovaním zobrazeného grafu filtrov na súbor SampleVideo_1280x720_10mb.mp4[3] sa v náhľade zobrazí:

Obrázok 2.10 znázorňuje vzťahy medzi implementovanými triedami. Z priestorových dôvodov sú vynechané atribúty a metódy. Pomocné triedy Utils, PipelineError a ParsedFilter taktiež nie sú znázornené.

[2] http://creativecommons.org/licenses/by-nd/3.0/
[3] Demonštračný súbor voľne dostupný z http://www.sample-videos.com/



Obr. 2.9: Náhľad na výstup filtrovania



Obr. 2.10: Vzťahy medzi triedami

Kapitola 3

Porovnanie s existujúcimi riešeniami

V tejto kapitole je vykonané porovnanie vytvorenej aplikácie s dvoma podobnými riešeniami pracujúcimi s grafmi filtrov - GraphEdit a Gst-Editor. Programy sú si síce príbuzné v rámci oblasti práce, ale vzhľadom na rozdielny použitý multimediálny framework je práca s nimi značne odlišná. V nasledujúcich častiach sa pokúsim aspoň porovnať spoločné rysy.

3.1 GraphEdit

GraphEdit pravdepodobne z dôvodu svojho veku nepodporuje niekoľko video formátov. Mnou bežne používaný testovací súbor SampleVideo_1280x720_10mb.mp4 sa mu nepodarilo otvoriť. Druhý testovací súbor clock.avi už síce otvoril, ale s poznámkou, že nie všetky obsahované streamy sú v podporovanom formáte. Po otvorení súboru zobrazí jeho vnútorný graf filtrov, s ktorým je možné manipulovať. Filtre sa do grafu pridávajú nie moc intuitívnym spôsobom, pri ktorom je najprv nutné zobraziť ponuku filtrov pomocou možnosti menu *Insert filters...*. Zo zobrazenej ponuky nie je možné filtre do grafu preťahovať, ale je nutné použiť tlačítko Insert filtre, ktoré zvolený filter pridá na ľavú stranu okna pod aktuálny graf filtrov. Plusom GraphEditu je, že všetky piny majú popisky s očakávaným tokom dát. Podporuje prácu s viacerými kanálmi a video aj audio stopami. Filtre sa spájajú ťahom myši z jedného pinu do druhého. V ponuke má veľké množstvo filtrov zaradených do rôznych kategórií. Vytvorený graf filtrov je možné prehrať pomocou možnosti Graph->Play. Z neznámeho dôvodu sa mi pri posúvaní okna prehrávača posúvalo aj prehrávané video.

3.2 Gst-Editor

Podobne ako GraphEdit aj Gst-Editor už má nejaký ten vek (posledná aktualizácia je z roku 2012) a bez úpravy zrojových kódov ho nie je možné na novších OS skompilovať. Nenašiel som formát, ktorý nedokázal otvoriť, ale pri práci s grafom filtrov opäť chýba možnosť pridávať filtre pomocou Drag and Drop. Ponuku dostupných filtrov má dostupnú bez nutnosti vyvolať akékoľvek menu. Filtre sú rozdelené do kategórií v stromovej štruktúre podľa využitia. Do grafu filtrov sa pridávajú dvojklikom a spájajú sa pomocou ťahu myši z jedného padu do druhého. Editor vlastností filtru je vo vlastnom okne a mimo upraviteľných vlastností ukazuje aj informácie o padoch. Gst-Editor prehrávač aktuálnej pipeline.

3.3 Porovnanie s Filter Graph Editor

V nasledujúcej tabuľke sú porvnané vyššie opísané nástroje s Filter Graph Editor.

Tabuľka 3.1: Porovnanie nástrojov pre prácu s grafom filtrov

	Filter Graph Editor	**GraphEdit**	**Gst-Editor**
Zoznam filtrov	v hlavnom okne	v dialógovom okne	v hlavnom okne
Vkladanie filtrov	drag and drop	kliknutím tlačítka	dvojklik
Podpora audio/video	len video	audio aj video	audio aj video
b	len jeden vstup a výstup	neobmedzene	neobmedzne
Podporované formáty	všetky podporované FFmpegom	malý počet, nutné doinštalovať	všetky podporované GStreamerom
Eitor vlastností filtru	v hlavnom okne	v dialógovom okne	vo vlastnom okne
Spájanie filtrov	kliknutím na spojovacie body	ťahom myši	ťahom myši
Podpora náhľadu výsledku	áno	áno	áno

Záver

Cieľom tejto práce bolo navrhnúť a implementovať program pre zostavovanie grafov filtrov v FFmpeg. Na úvod bol čitateľ oboznámený s pojmami z oblasti spracovania videa a obrazu a oboznámený s multimediálnymi frameworkmi pracujúcimi s grafom filtrov. V popise spomínaných multimediálnych frameworkov bolo poukázané na rôznu používanú terminológiu pri práci s grafmi filtrov. V nasledujúcej kapitole boli prebrané návrhy a implementácia programu.

Grafické používateľské rozhranie bolo vyvýjané pomocou frameworku Qt a backend je napísaný za pužitia knižníc FFmpegu. Celý vývoj prebiehal v operčnom systéme Linux Mint. Vďaka multiplatformnosti Qt aj FFmpegu by mala aplikácia bežať pod väčšinou operačných systémov, pre ktoré sú Qt a FFmpeg dostupné. V priebehu vývoja som vytvoril zopár projektov fungujúcich nezávisle od seba poskytujúce rôznu funkcionaliu. Výsledný program bol zostavený tak, aby dokázal pracovať s ľubovoľným grafom filtrov a výsledok spracovania dokázal buď zobraziť v okne alebo ho zapísal do súboru. Okrem spracovania sa do výsledného programo podarilo implementovať ukladanie a načítanie grafu filtrov a import a export reťazca pre FFmpeg nástroj ffplay. Výsledný program bol testovaný na voľne dostupných videách v rôznych rozlíšeniach a formátoch.

Najväčším problémom pri vyhotovovaní riešenia bola práca s FFmpegom, ku ktorému existuje veľmi málo tutoriálov alebo príkladov, prípadne ak existujú, sú zastaralé a nepltné pre aktuálnu verziu FFMpegu. Ďalším problémom FFmpegu je značne nedostačujúca dokumentácia a zdrojové kódy postrádajúce komentáre. Rovnako sa nedá povedať, že by tieto nekomentované zdrojové kódy boli jednoduché na pochopenie.

Behom práce som dostal niekoľko nápadov, ktoré by mohli byť realizovateľné v budúcnosti. Ako sa možno dočítať v tomto dokumente, implementovaná aplikácia pracuje len s video stopami. Zahrnutie práce s audio stopami by mohol byť jeden zo smerov, ktorým by sa vývoj uberal ďalej v budúcnosti. Taktiež je program limitovaný na jeden jediný vstup a výstup. Preto by ďalším možným rozšírením mohla byť práca s viacerými súbormi alebo kanálmi. A nakoniec by aplikácii pomohlo, keby obsahovala plnohodnotný prehrávač náhľadu s možnosťou pozastavenia a znovuspustenia.

Literatúra

[1] Bařina, D.; Zemčík, P.: *Multimédia*. 2013.

[2] Communications Museum of Macao: *Multiplexing and Demultiplexing*. [online]. 2016, [cit. 21.1.2016].
URL http://macao.communications.museum/eng/exhibition/secondfloor/MoreInfo/2_8_6_Multiplexing.html

[3] Ffmpeg.org: *FFmpeg*. [online]. 2016, [cit. 21.1.2016].
URL https://www.ffmpeg.org

[4] Freedesktop.org: *Building an Application*. [online]. 2016, [cit. 20.1.2016].
URL http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/part-building.html

[5] Libav.org: *Libavfilter Documentation*. [online]. 2016, [cit. 14.5.2016].
URL https://libav.org/documentation/libavfilter.html

[6] Microsoft: *About DirectShow Filters*. [online]. 2016, [cit. 20.1.2016].
URL
https://msdn.microsoft.com/en-us/library/windows/desktop/ms696274.aspx

[7] Microsoft: *About Media Foundation*. [online]. 2016, [cit. 20.1.2016].
URL
https://msdn.microsoft.com/en-us/library/windows/desktop/ms696274.aspx

[8] The Qt Company: *About Qt*. [online]. 2016, [cit. 12.5.2016].
URL https://wiki.qt.io/About_Qt

[9] The Qt Company: *Phonon multimedia framework*. [Online]. 2016 [cit. 20.1.2016].
URL http://doc.qt.io/qt-4.8/phonon-overview.html

[10] The Qt Company: *Qt Widgets*. [online]. 2016, [cit. 14.5.2016].
URL http://doc.qt.io/qt-5/qtwidgets-index.html

[11] The Qt Company: *Signals & Slots*. [online]. 2016, [cit. 12.5.2016].
URL http://doc.qt.io/qt-5.6/signalsandslots.html

[12] Španěl, M.: *Základy počítačové grafiky: Úvod do predmětu*. 2015.

Prílohy

Zoznam príloh

Príloha A

Obsah CD

- doc/ - priečinok s dokumentáciou vygenerovanou Doxygenom
- examples/ - priečinok obsahujúci testovacie videá a uložené grafy filtrov
- src/ - Priečinok so zdrojovými kódmi aplikácie
- text/ - Priečinok so zdrojovými kódmi technickej správy
- Doxygen - Nastavenie Doxygen pre generovanie dokumentácie
- report.pdf - Táto správa
- Makefile - Makefile pre skompilovanie a spustenie programu, vygenerovanie dokumentácie a výpisu metrík kódu
- README - Súbor s informáciami o spustení a testovaní aplikácie