VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

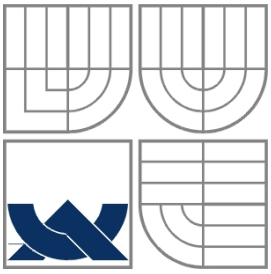
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

EDITOR VIDEA

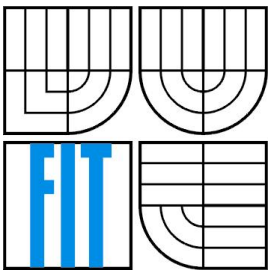
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE PETR DRASTIL
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

EDITOR VIDEA
VIDEO EDITOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR DRASTIL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ROMAN JURÁNEK

BRNO 2010

Abstrakt

Obsahem práce je návrh a implementace jednoduché aplikace pro zpracování a střih videa. Práce rozebírá základní funkce programu: načtení video sekvence, střih, vložení na časovou osu a vytvoření výstupní video sekvence. Jedná se o aplikaci s grafickým uživatelským rozhraním, které je určené pro uživatele amatéra. Teoretická část práce se zabývá srovnáním současných prostředků pro zpracování video sekvencí a nástroji pro vytváření uživatelských rozhraní.

Abstract

This work deals with design and implementation of a simple application for video processing and video editing. Work examines essential functions of program: video sequence import, sequence cutting, timeline placement and video sequence export. This software has graphical user interface which is intended for amateur user. Theoretical part of this thesis describes current software for video editing and tools for creation of graphical user interfaces.

Klíčová slova

video, střih videa, film, nelineární střih videa, zpracování videa, Qt Toolkit, GUI, grafické uživatelské rozhraní

Keywords

video, video editing, nonlinear editing, video processing, Qt Toolkit, GUI, graphical user interface

Citace

Petr Drastil: Editor videa, bakalářská práce, Brno, FIT VUT v Brně, 2010

Editor Videa

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Romana Juránka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Drastil
1.5.2010

Poděkování

Tímto bych chtěl poděkovat Ing. Romanu Juránkovi za odbornou pomoc při tvorbě praktické a teoretické části mé práce.

© Petr Drastil, 2010
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod

Člověk je v dnešní době obklopen mnoha druhy multimédií. Nejvíce se však prosazuje digitální video, s kterým se setkáváme na každém kroku v podobě televizního vysílání, filmové projekce v kině, na obrazovkách našich počítačů nebo na displejích přenosných zařízení. Díky poklesu cen video techniky má dnes již každý možnost pořídit si kameru a stát se hercem vlastního filmu. Tento trend je navíc podporován různými službami na internetu, které nabízejí uživateli možnost sdílet vlastní vytvořené filmy. Zpracování natočeného videa bylo donedávna výsadou profesionálních studií. S rozšířením počítačové techniky v běžných domácnostech však došlo k obratu, a tak si dnes i běžný uživatel může upravit vlastní natočené video a vytvořit tak zajímavý sestřih záběrů ze své dovolené.

Cílem mé práce bylo vytvoření jednoduchého programu pro editaci videa, který by uživateli nabídl základní paletu nástrojů pro práci s videem. Zároveň by program měl mít dostatečně intuitivní ovládání, aby ho zvládl ovládat i amatérský uživatel, který s podobnými programy nemá prozatím zkušenost.

Úvodní kapitola mé práce se zaměřuje na přehled dnešních video editorů a popisuje jejich základní vlastnosti. Další kapitola se věnuje použité knihovně pro grafické uživatelské rozhraní, které bylo zvoleno pro vývoj naší aplikace. Druhá část této kapitoly zavádí běžně používanou terminologii a na příkladech demonstruje specifické vlastnosti použité knihovny. Kapitola 4 se věnuje jádru bakalářské práce. Začátek této kapitoly popisuje použité implementační prostředky. Další část se zaměřuje na práci s vytvořeným programem z pohledu uživatele a zbytek kapitoly popisuje implementaci jádra a jednotlivých komponent, ze kterých je program Video Editor sestaven. Závěrečná kapitola se věnuje omezením programu a budoucím možným rozšířením.

2 Střihové programy

Programů pro zpracování videa lze nalézt na dnešním trhu celou řadu. Hlavní rozdělení těchto programů je dle prostředí, ve kterém budou použity. Záleží tedy na tom, zda jsou určeny pro profesionální účely nebo pro zpracovávání domácího videa. Jednotlivé programy se liší především počtem nástrojů, efektů a v neposlední řadě i cenou.

Z řad profesionálních video editorů lze jmenovat Adobe Premiere Pro, Sony Vegas Pro nebo Apple Final Cut. Tyto programy podporují editaci videa ve velmi vysoké kvalitě, umožňují jemné úpravy obrazu a zvuku a obsahují širokou škálu efektů, filtrů a přechodů, které může uživatel aplikovat na výsledné video. Cena těchto profesionálních nástrojů se pohybuje mezi 600 až 1000 dolary. Alternativou ke komerčním střihovým programům je pak skupina open source projektů, mezi které patří programy jako Kdenlive, VirtualDub nebo OpenShot, které se svou užitnou hodnotou do jisté míry profesionálním nástrojům vyrovnávají.

Grafické uživatelské rozhraní všech zmiňovaných editorů se skládá z několika stejných prvků. Každý editor obsahuje okno s časovou osou, okno pro přehrávání náhledu upravovaného videa, nebo přehrávání náhledu scénáře a okno, které obsahuje veškeré zdrojové soubory. Podrobnější popis těchto ovládacích prvků lze nalézt v kapitole 4.1, která se věnuje implementaci Video Editoru.

2.1 Adobe Premiere Pro

Adobre Premiere je profesionální nástroj pro nelineární střih videa, který je dodávaný jako součást balíku Adobe Creative Suit. Jedná se o jeden z nejznámějších programů pro zpracování videa a je podobně úspěšný jako Adobre Photoshop pro zpracování fotografií. Jeho první verze byla vydána v prosinci roku 1991 pro operační systém Mac OS. O dva roky později byl tento program přenesen i na platformu Windows [1]. Grafické uživatelské rozhraní toto programu je zobrazeno na obrázku 2.1.

Obrázek 2.1: Grafické uživatelské rozhraní programu Adobe Premiere Pro

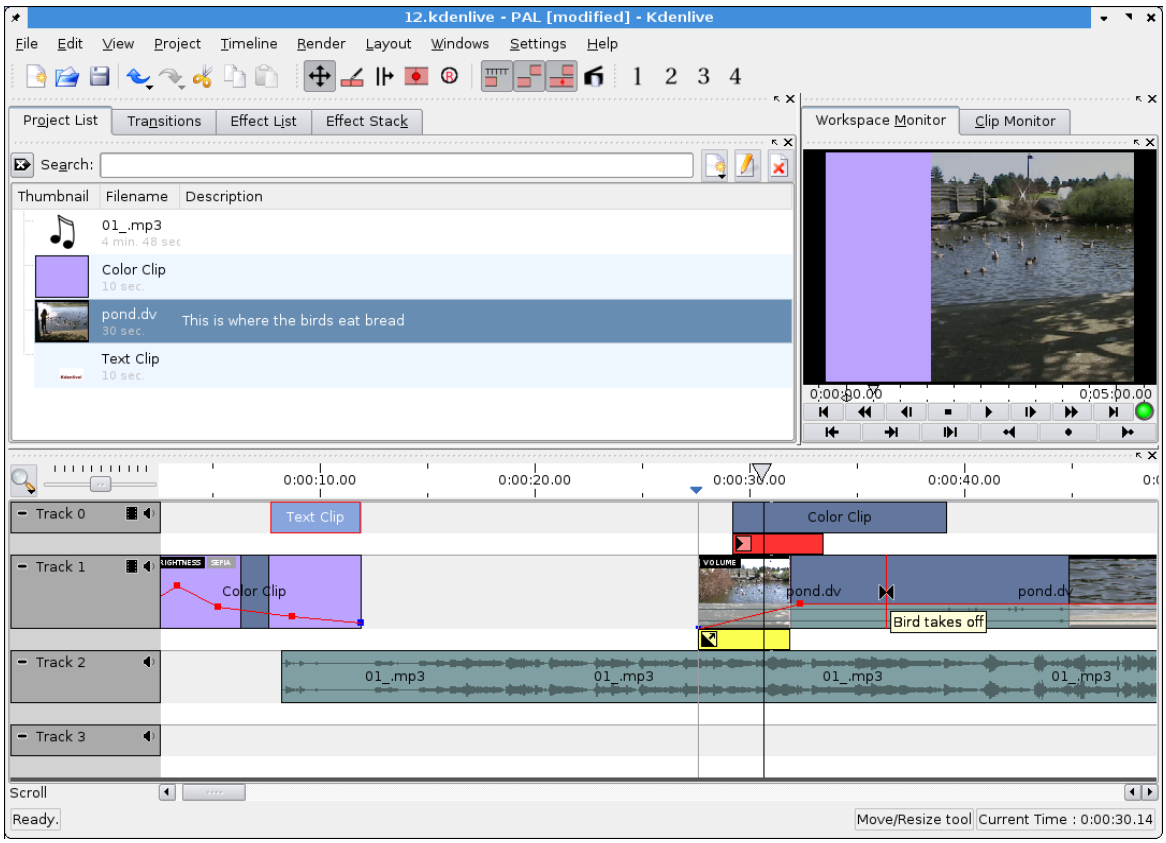
Program se distribuuje ve dvou verzích, které se liší nabízenými nástroji a cenou. Adobe Premiere Pro je plnohodnotná verze určená pro profesionální použití, která nabízí rozsáhlou nabídku nástrojů, efektů, filtrů, širokou podporu formátů pro obraz a video. V aktuální verzi, která se má objevit na trhu 29. dubna 2010, má být nástroj Mercury PlayBack Engine, který umožňuje plynulou práci s HD videem pomocí nativních 64-bitových výpočtů pomocí GPU. Toto je významný krok kupředu, jelikož dosud nebylo možné pracovat s videem ve vysokém rozlišení bez značných časových prodlev. Profesionální verze byla použita například při vytváření filmů „Superman se vrací“, „Kapitán Abu Raed“ a „Dust to Glory“. Mimo to je dlouhodobě používána zpravodajskou stanicí BBC.

Od profesionální verze je odvozena verze Adobe Premiere Element, která je určena pro běžného uživatele. Hlavním rozdílem je, že nedovoluje renderování výsledného videa ve vysoké kvalitě, neposkytuje nástroje pro dodatečnou úpravu barevného vyvážení obrazu a nedovoluje editaci z více video kamer najednou. Největším rozdílem je však cena, která je oproti profesionální verzi desetinová.

2.2 Kdenlive

Kdenlive je nelineární editor pro platformu GNU/Linux, FreeBSD a Mac OS. Jedná se o open source projekt, který v roce 2002 odstartoval Jason Wood. Nyní na Kdenlive pracuje malý tým vývojářů a tým dobrovolníků.

Jádro editoru je postaveno na knihovně FFMpeg [10] a multimediálním frameworku MLT (Media Lovin’ Toolkit), který poskytuje sadu nástrojů pro práci se záběry, stopami, efekty a přechody. Knihovna FFMpeg umožňuje práci s audio a video formáty jako jsou MOV, AVI, WMV, MPG, XviD a další. Grafické uživatelské rozhraní je zobrazeno na obrázku 2.2.

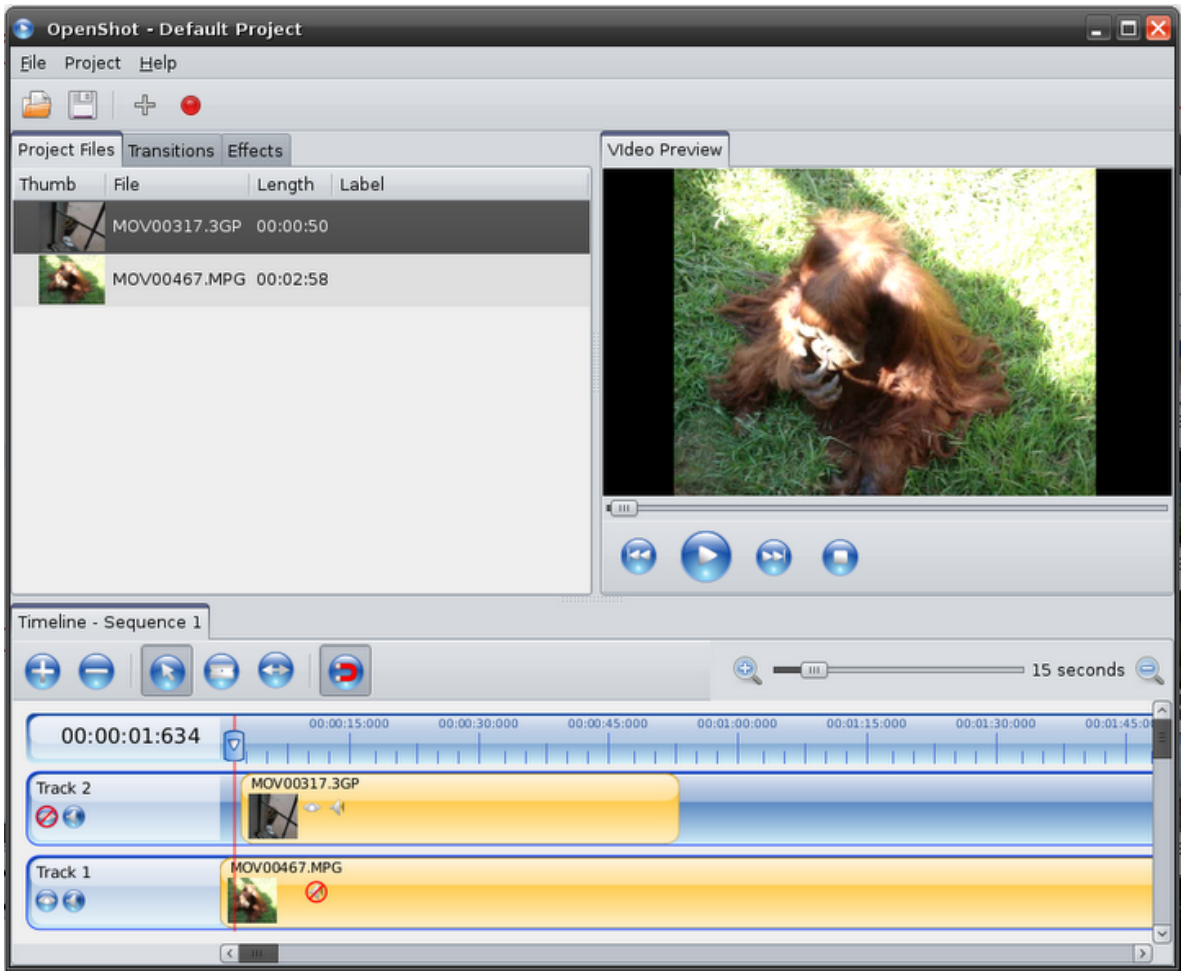


Obrázek 2.2: Grafické uživatelské rozhraní programu Kdenlive

Program umožňuje editaci videa na více stopách, přičemž množství audio a video stop není omezeno. Mimo editace je podporováno i nahrávání videa z obrazovky, přes Firewire i přes V4L (webové kamery, TV karty). Renderování výsledného videa je možné spustit na pozadí a při tom dále pracovat na projektu. Druhou možností je uložení výsledného projektu do souboru pro dávkové zpracování. Program obsahuje spoustu nástrojů pro práci s efekty, přechody, nástroje pro vytváření a vkládání titulků do videa a nástroje pro vytváření DVD disků včetně menu. Informace o tomto programu byly čerpány z [2], [3] a [4], kde je možné dozvědět se více.

2.3 OpenShot

OpenShot je další z řady open source video editorů. Jeho vývoj započal v roce 2008 Jonathanem Thomasem. Aktuálně na programu pracuje 11 vývojářů a spousta dobrovolníků. Aplikace je napsána pomocí skriptovacího jazyku Python s pomocí knihoven MLT a FFMpeg. Grafické uživatelské rozhraní programu je zobrazeno na obrázku 2.3.



Obrázek 2.3: Grafické uživatelské rozhraní programu OpenShot

Program podporuje práci na více časových stopách, umožňuje zvětšovat, zmenšovat a ořezávat video, přidávat efekty a zobrazovat celkové náhledy výsledného videa. Aplikace též umožňuje pracovat se statickým obrazem, který je uložen v bitmapovém a vektorovém formátu. Nástroje pro práci s obrazem umožňují přidávat vodoznaky nebo prokládat různé obrázky do sebe. Pro práci se zvukem jsou k dispozici nástroje, které umožňují snadno složit zvuk z několika stop do jednoho výsledného kanálu. Další informace o programu lze nalézt na [5].

3 Knihovny pro grafické uživatelské rozhraní

Pro vytvoření uživatelského rozhraní musí být zvolen vhodný toolkit, který poskytne nástavbu nad programovacím jazykem C++. Mezi nejznámější a nejpoužívanější frameworky patří Qt Toolkit, GTK+ a wxWidgets. Kapitola 3.1 popisuje historii a vlastnosti Qt Toolkitu, který byl vybrán pro tvorbu uživatelského rozhraní a kapitola 3.2 popisuje specifika vývoje v tomto frameworku a dále stanoví běžně používanou terminologii.

3.1 Qt Toolkit

Qt [6] patří mezi jednu ze dvou nejpopulárnějších multiplatformních knihoven pro vytváření programů s grafickým uživatelským rozhraním. Od verze 4 lze vytvářet i konzolové aplikace, které využívají výhod této knihovny. Tento framework byl dříve vyvíjen společností Trolltech, která ho následně prodala společnosti Nokia a ta ho udržuje do dnes. Největší popularitu získal Qt Toolkit přibližně před dvěma lety, kdy bylo uvedeno nové prostředí KDE 4, které je implementováno jen s použitím této knihovny. Mezi další velice úspěšné projekty, které byly napsány s pomocí tohoto frameworku, patří například Opera, Google Earth, Skype a VirtualBox.

Qt Toolkit poskytuje širokou škálu knihoven, které sahají od generických algoritmů až po knihovny pro práci s multimédii. Knihovny jsou udržovány jak společností Nokia, tak i širokou komunitou, takže pravidelně vycházejí nové verze, které obsahují opravy a nové nástroje. Společnost Nokia navíc nabízí vlastní integrované vývojové prostředí QtCreator, které obsahuje kvalitní integrovanou fulltextovou nápovědu, podporu správy kódu Git a Subversion a spoustu dalších nástrojů. Mimo jiné jsou k dispozici i pluginy pro integraci ve vývojových prostředích Microsoft Visual Studio a Eclipse. Všechny nástroje jsou k dispozici zdarma pro open source projekty. Pro komerční použití se musí zakoupit licence.

Framework podporuje vývoj pro mnoho druhů platforem. Aplikace lze vyvíjet pro operační systémy Microsoft Windows, Linux, Mac OS nebo pro různé vestavné operační systémy. Díky použití Meta Object Compileru (MOC) je zaručena 100% přenositelnost mezi systémy bez nutnosti měnit jakoukoliv část zdrojových souborů. MOC není kompilátor, přestože by tomu název nasvědčoval. Tento nástroj prozkoumává vytvořený zdrojový kód před kompilací a vyhledává klíčová slova platná pro Qt Toolkit, která jsou následně nahrazena jiným zdrojovým kódem optimalizovaným pro cílovou platformu. Framework též využívá API cílové platformy k tomu, aby uzpůsobil chování a vzhled svých komponent tak, aby výsledná aplikace vypadala na dané platformě nativně. Knihovnu je možné používat kromě jazyka C++ i v jazycích Python, Java a dalších.

3.2 Úvod do programování v Qt Toolkitu

Základním pojmem pro Qt Toolkit je *widget*, což je vizuální element uživatelského rozhraní. Tento termín vznikl z „window gadget“, který se používá v operačním systému Unix/Linux a je ekvivalentem pro „control“ a „container“ v terminologii Windows. Příkladem widgetu mohou být tlačítka, různá menu, dialogová okna a další komponenty. Widget může být vytvořen buď od bázové třídy `QWidget` a nebo může být složen kompozitním způsobem z již existujících widgetů. Pro aplikační okno je nejčastěji použita třída `QMainWindow` nebo `QDialog`. Velká flexibilita Qt Toolkitu však

umožňuje, aby aplikační okno bylo tvořeno jakoukoliv dostupnou komponentou, což je demonstrováno na příkladu níže.

Základním stavebním kamenem frameworku je třída `QObject`, která implementuje návrhový vzor „Observer“. Ten řeší problém, kdy je definovaná závislost jednoho objektu na druhém. Závislost, ve smyslu návrhového vzoru, představuje propagaci změny nezávislého objektu závislým objektům (pozorovatelům). Snahou tohoto přístupu je oddělení nezávislých nadřízených objektů od závislých objektů tak, aby tyto objekty mohly být informovány bez znalosti vnitřní struktury nadřízeného objektu. V terminologii Qt Toolkitu je implementace tohoto návrhového vzoru označována pojmy signál a slot.

Termín *emitovat signál* označuje propagaci změny pozorovatelům. Funkce pozorovatele, která přijímá změnu, je nazývána *slotem*. Pro správnou funkcionalitu tohoto mechanismu není podporována vícenásobná dědičnost, jelikož všechny třídy mají za předka třídu `QObject`. Princip práce se signály je vysvětlen na následujícím příkladu, který je převzatý z [7].

```
 1 #include <QApplication>
 2 #include <QPushButton>

 3 int main(int argc, char *argv[])
 4 {
 5     QApplication app(argc, argv);
 6     QPushButton *button = new QPushButton("Quit");
 7     QObject::connect(button, SIGNAL(clicked()),
 8                      &app, SLOT(quit()));
 9     button->show();
10     return app.exec();
11 }
```

Cílem tohoto příkladu je vytvořit hlavní okno, které bude obsahovat jedno tlačítko. Pokud dojde ke stisku tlačítka uživatelem, tak se aplikace ukončí. K vytvoření aplikace jsou zapotřebí dvě třídy. První se nazývá `QApplication` a slouží jako základ všech aplikací napsaných v Qt Toolkitu, jelikož se stará o správu přidělených zdrojů a chod uživatelského rozhraní. Druhá třída se nazývá `QPushButton` a reprezentuje požadované tlačítko, které bude sloužit jako aplikační okno.

Propojení aplikace s tlačítkem je realizováno na řádku 7 pomocí statické funkce connect() třídy `QObject`. Makra SIGNAL() a SLOT() jsou součástí syntaxe. Tento řádek říká, že signál clicked(), který se emituje při stisknutí tlačítka, automaticky volá slot quit(), který aplikaci ukončí. Na řádku číslo 9 je tlačítko zobrazeno a na řádku 10 dojde ke spuštění hlavní aplikační smyčky. Výsledná podoba aplikace je na obrázku 3.1.

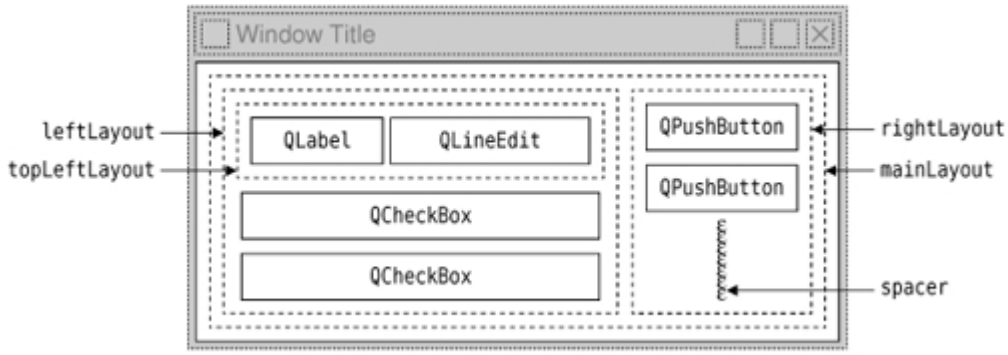
Obrázek 3.1: Jednoduchá aplikace v Qt Toolkitu, převzato z [7]

Na jeden signál je možné pomocí funkce connect() namapovat více slotů. V takovém případě není předem známo, v jakém pořadí budou tyto sloty volány. Mimo jiné se na jeden signál dají namapovat další signály, což je výhodné v případě vytváření uživatelského kompozitního widgetu, kdy je nutné propagovat signál z vnitřních objektů do vnějšího zapouzdřujícího objektu.

Pro implementování signálů a slotů uvnitř uživatelské třídy je důležité použít makro Q_OBJECT. Toto makro říká Meta Object Compileru, že se má k této třídě chovat jinak než

ke klasické třídě v C++, protože bude používat mechanismus signálů a slotů. V příkladě není toto makro nikde použito, jelikož ho všechny použité třídy obsahují.

Poslední důležitou součástí Qt Toolkitu jsou layouty. To jsou neviditelné mřížky, které sdružují jednotlivé widgety v určitých ucelených blocích. Tyto bloky mohou být orientovány horizontálně, vertikálně a do mřížky. Každý layout může zároveň obsahovat další vnořené layouty. Výhoda layoutů spočívá v automatické změně velikosti všech ovládacích prvků při zachování jejich rozmístění uvnitř widgetu v případě, že dojde ke změně velikosti aplikačního okna. Toto je demonstrováno na obrázku 3.2.



Obrázek 3.2: Použití layoutů převzato z [8]

Layoutu jsou na obrázku znázorněny jako přerušované čáry. Při změně velikosti hlavní okna dojde ke změně vnějšího layoutu, který propaguje tuto změnu do vnitřních layoutů. Pružina v pravém dolním rohu znázorňuje prostor, který bude vždy obsahovat prázdné místo.

4 Návrh a implementace video editoru

Tato kapitola se věnuje jádru bakalářské práce, které představuje návrh architektury a implementaci programu pro nelineární střih videa. Hlavním cílem práce bylo vytvoření uživatelského rozhraní, které by poskytovalo uživateli intuitivní ovládání a umožňovalo mu rychlou práci s videem. Úvodní část této kapitoly je věnována implementačním prostředkům a základnímu popisu práce s video editorem. Další části se zaměřují na návrh a implementaci jednotlivých ovládacích komponent, které tvoří základní kameny tohoto projektu.

Aplikace Video Editor umožňuje:

- nelineární střih videa
- práci se statickým obrazem
- přehrávání náhledu aktuálně upravovaného multimédia
- přehrávání video sekvencí z časové osy
- možnost ukládat a načítat uživatelské projekty
- export výsledného scénáře do video souboru

Program je určen především pro uživatele amatéra, který nechce složité uživatelské rozhraní s mnoha pokročilými funkcemi, které nabízejí profesionální nástroje. Veškeré otevřené a editované soubory představují *projekt*, který si uživatel může uložit do souboru s příponou *.vep (video editor projekt). Tento konfigurační soubor má XML strukturu a obsahuje informace o otevřených souborech, *sekvencích* vytvořených z těchto souborů a jejich umístění na časových stopách. Výsledná sekvence složená z časových stop představuje *scénář* projektu, který lze přehrát nebo exportovat do nového video souboru. Konfigurační soubory lze do budoucna použít i nějakou konzolovou aplikací, která bude využívat jádra video editoru a dvou tříd pro jejich parsování, což umožní hromadné zpracování projektů.

4.1 Implementační prostředky

Aplikace Video Editor je implementována v jazyce C++, bylo využito principů objektově orientovaného programování. Pro veškerou práci s obrazem byla použita multiplatformní knihovna OpenCV (Open Source Computer Vision) [9].

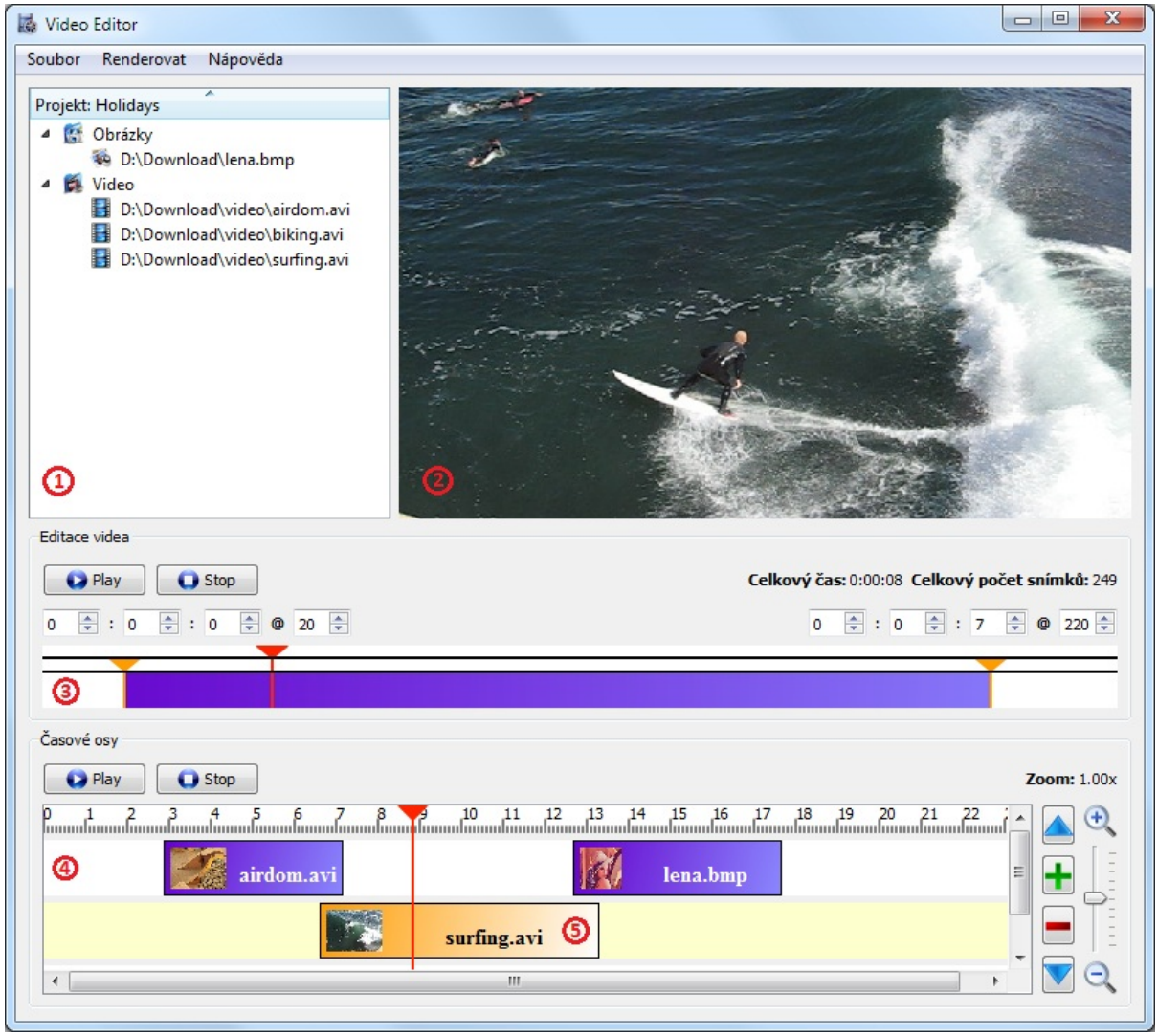
Tato knihovna byla vyvinuta pro zpracování problémů počítačového vidění. Knihovna byla zvolena, protože je volně dostupná a poskytuje širokou škálu vysoce optimalizovaných funkcí pro zpracování obrazu a videa. Úspěšný překlad programu je podmíněn instalací knihovny OpenCV 2.0 a vyšší.

Pro grafické uživatelské rozhraní byl zvolen již dříve jmenovaný Qt Toolkit. Podmínkou pro správný běh je přeložení této knihovny s podporou XML a OpenGL. Minimální verze Qt Toolkitu musí být 4.6.1 a vyšší.

Aplikace byla vyvíjena a laděna pod operačním systémem Microsoft Windows. Kód programu je přenositelný i na operační systém GNU/Linux.

4.2 Hlavní okno video editoru

Hlavní okno se skládá z okna projektu, okna monitoru, okna pro ořezávání videa a časové osy. Vzhled aplikačního okna je dán layoutem, který zachovává pevnou šířku okna s projektem a pevnou výšku okna pro ořezávání videa. Zbylé widgety se dynamicky mění podle toho, jak si uživatel okno roztáhne, přičemž nejvíc prostoru si vždy zabere okno s časovou osou. Hlavní okno je implementováno ve třídě `Videoeditor`, která se stará o propojení všech widgetů pomocí patřičných signálů a slotů. Pracovní plocha je zobrazena na obrázku číslo 4.1. Čísla, která se na obrázku nacházejí v kroužcích, slouží pro referenci v dalších podkapitolách. V textu budou tyto reference označeny jako (1), (2), atd.

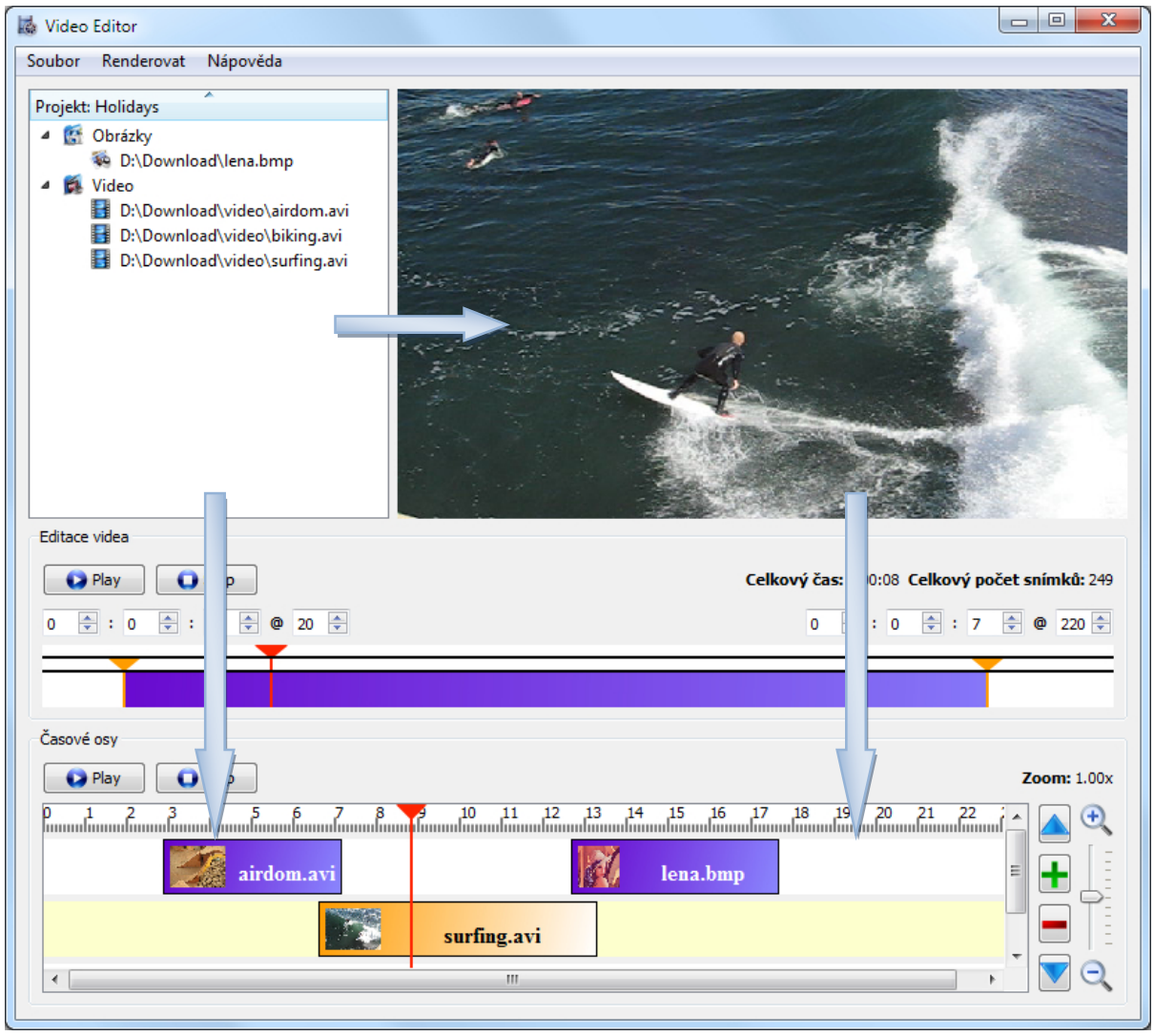


Obrázek 4.1: Pracovní plocha video editoru

4.2.1 Okno projektu

Okno (1) zobrazuje a organizuje zdrojový materiál. Aktuálně jsou k dispozici obrázky a video bez audio stopy. Záběry jsou do projektu načteny ze souborů. Přetažením záběru na časovou osu se stane záběr součástí vytvářeného scénáře. Přetažením záběru do okna monitoru se vytvoří pracovní kopie,

kterou můžeme nejdříve editovat, než jí přidáme na časovou osu. Veškeré možné způsoby přetažení multimediálních souborů jsou znázorněny na obrázku číslo 4.2 pomocí šipek.



Obrázek 4.2: Možnosti přetažení multimediálních souborů

4.2.2 Okno monitoru

Slouží pro úpravu a náhled načtených záběrů. Chování okna (2) záleží na akci uživatele. Buď může zobrazovat náhled při umisťování počátečního a koncového bodu střihu v okně pro ořezávání videa (3) nebo může zobrazovat celkový náhled scénáře z časové osy, která se nachází ve spodním okně (4). Uživatel může mimo jiné vytvořit kopii i z aktuálně zobrazeného záběru v monitoru jeho tažením do okna časové osy. V případě, že není vytvořena pracovní kopie nebo se nepracuje s vybranou sekvencí v časové ose, nelze přetažení z monitoru provést.

4.2.3 Okno pro ořezávání videa

Toto okno umožňuje uživateli umístit počáteční a koncový bod střihu ve videu, přičemž pro nastavení je k dispozici více možností. Prvním způsobem je navolení počátečního a koncového bodu pomocí sady spinboxů, které se nacházejí nalevo a napravo a slouží zároveň k zobrazení číselné informace uživateli. Hodnotu bodu střihu lze pomocí spinboxů navolit nahrubo pomocí času nebo najemno pří-

mo na konkrétní snímek ve videu. Druhá možnost nastavení počátečního a koncového bodu spočívá v použití widgetu (3), který má ve spodní části dva jezdce určené pro tuto operaci. Horní část komponenty obsahuje dalšího jezdce, kterým uživatel ovládá přehrávací hlavu aktuální editované sekvence. Při manipulaci s přehrávací hlavou je vždy v monitoru zobrazován aktuální snímek určený pozicí jezdce přehrávací hlavy, i když zrovna nedochází k přehrávání náhledu celé editované sekvence.

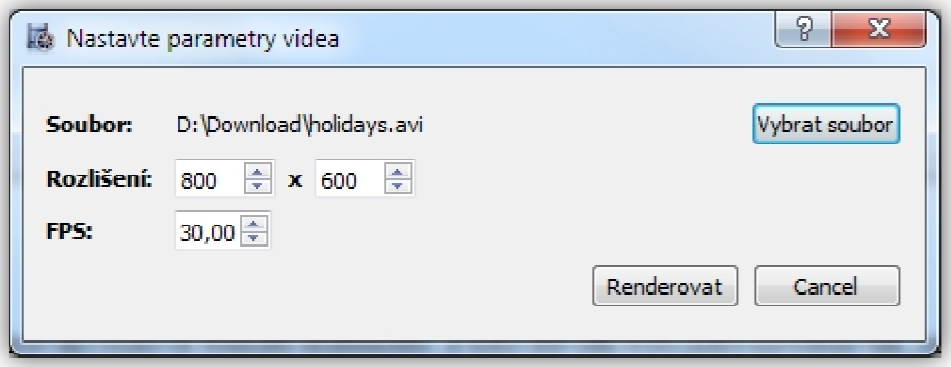
4.2.4 Okno časové osy

Okno obsahuje časové stopy (4), na které jsou nanášeny načtené a upravené sekvence. Cílem úprav na časové ose je uspořádat sekvence tak, aby tvořili výsledný scénář. Počet stop není nijak omezen. Stopa, která je nejvýše v celé hierarchii, má vždy prioritu před nižší stopou. To znamená, že pokud jsou dvě sekvence souběžně umístěny do různých stop a zároveň se překrývají, bude záběr z vyšší stopy vždy na popředí. Umístěním dvou sekvencí za sebou do stejné stopy dojde k jejich spojení. Přechody mezi sekvencemi jsou realizovány pomocí ostrého střihu.

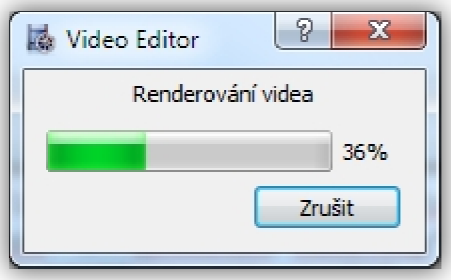
V horní části okna se nachází časové pravítko, které uživateli poskytuje informaci o aktuální zobrazené části scénáře a aktuální pozici přehrávací hlavy pro náhled výsledného scénáře. Přehrávací hlava je ovládána pomocí jezdce v časovém pravítku nebo kliknutím do oblasti časových stop. Pokud by uživatel potřeboval ke své práci větší nebo menší časové měřítko, lze časovou osu přiblížit nebo oddálit pomocí jezdce, který se nachází v pravé části okna. Tuto operaci lze vykonat ještě pomocí kolečka myši nebo pomocí klávesnice. Výhodou a charakteristickým rysem nelineárního střihu je možnost vložit záběr i dodatečně mezi záběry ostatní, přičemž záběry za místem vložení jsou odsunuty doprava. Uživatel je informován o posunutí pomocí animace, která slouží jako vizualizace daného procesu.

4.2.5 Okno pro renderování videa do souboru

Toto okno je vyvoláno z nabídky v menu a umožňuje uživateli zadat parametry výstupního video souboru, který je vytvořen ze scénáře. Vzhled okna je vidět na obrázku 4.3. Po potvrzení parametrů vytvářeného video souboru dojde k zavolání renderovací metody komponenty časové osy a výsledné snímky jsou předávány třídě `VideoWriter`, která snímky upraví na cílové rozlišení pomocí lineární interpolace a následně je uloží do souboru. Celý průběh operace lze sledovat v modálním dialogovém okně, které dovoluje danou akci přerušit. Toto okno je zobrazeno na obrázku číslo 4.4.



Obrázek 4.3: Dialog s možnostmi pro nastavení parametrů výstupního videa



Obrázek 4.4: Zobrazovaný průběh při renderování videa

4.2.6 Ukládání a nahrávání projektu

Celý uživatelský projekt lze uložit do XML konfiguračního souboru. Ten má následující tvar:

```
<?xml version="1.0" encoding="UTF-8"?>
<videoeditor>
   <sources>
     <source path="D:/Download/lena.bmp"/>
     <source path="D:/Download/video/airdom.avi"/>
   </sources>
   <timelinetracks>
      <track>
         <object path="D:/Download/video/airdom.avi" start-
         Time="2.86667" stopTime="7.16667" startFrame="1" stopFra-
         me="130" totalFrames="130" frameRate="30.0003"/>
          <object path="D:/Download/lena.bmp" startTime="12.6667"
         stopTime="17.6667" startFrame="1" stopFrame="150" to-
         talFrames="150" frameRate="30"/>
      </track>
      <track>
   </timelinetracks>
</videoeditor>
```

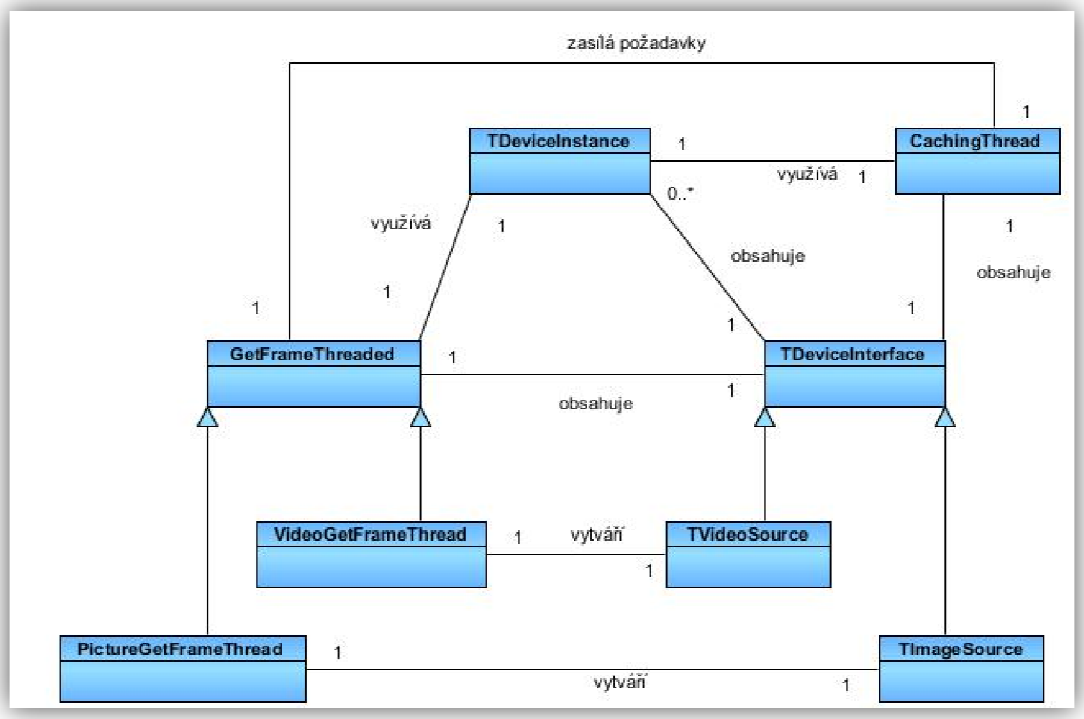
Sekce `<sources>` obsahuje veškeré otevřené zdroje v projektu, které jsou vidět v projektovém okně. Sekce `<timelinetracks>` obsahuje informace o časových stopách. Každá stopa je reprezentována tagem `<track>`, který již nemusí obsahovat další podsekce, což znamená, že daná stopa neobsahuje žádné umístěné sekvence. Umístění časových stop v okně časové osy záleží na jejich pořadí v souboru a to tak, že dříve nalezená stopa je v časové ose umístěna nejvýše. Každá stopa může dále obsahovat jednotlivé sekvence scénáře reprezentované tagem `<object>`, který obsahuje potřebné atributy pro umístění do časové stopy. Názvy jednotlivých atributů jsou samo popisné, a proto se jimi nebudeme zabývat.

Pro načítání a ukládání projektu do souboru slouží dvě třídy `XmlStreamReader` a `XmlStreamWriter`, které jsou napsány tak, aby bylo možné projekt kdykoliv rozšířit a přidat jen patřičnou metodu pro nově zavedený blok. Navíc je zajištěna i zpětná kompatibilita se staršími verzemi, jelikož neznámé tagy, které leží uvnitř bloku `<videoeditor>` se přeskakují, takže pokud nedojde k radikální změně celé struktury lze používat projekty z novějších verzí i ve starších verzích Video Editoru. Během načítání projektu dochází ke kontrole zdrojových souborů. Pokud soubory neexistují nebo byly přesunuty, tak dochází k ignorování patřičných tagů <object> a na konci načítání

je uživatel informován dialogovým oknem o možných problémech. V případě, že chce uživatel načíst nový projekt nebo ukončit Video Editor s rozpracovaným projektem, který byl nějak modifikován nebo nebyl uložen, tak dochází k vyzvání uživatele patřičným dialogem, který ho nechá vybrat další akci. Nabízené operace jsou uložení projektu, zahození projektu a zrušení volané operace.

4.3 Jádro video editoru

Pro správu a přehrávání jednotlivých souborů video editorem bylo potřeba vytvořit jádro, které by bylo dostatečně abstraktní a zároveň jednoduše rozšířitelné o další třídy, které by podporovaly přehrávání různých formátů multimédií. Z toho důvodu bylo jádro navrženo tak, jak je zobrazeno na diagramu tříd níže.



Obrázek 4.5: Diagram tříd znázorňující propojení jádra video editoru

Základní reprezentací multimediálního souboru zprostředkovává třída `TDeviceInterface`, která obsahuje čistě virtuální funkce pro přehrávání, zastavení, pozastavení a posun čtecí hlavy v daném multimediálním souboru. Navíc obsahuje funkce pro vytváření, vyhledávání a mazání instancí třídy `TDeviceInstance`, která reprezentuje jednu umístěnou sekvenci ve scénáři. Nad třídou `TDeviceInterface` je implementována třída `TVideoSource`, která umožňuje přehrávání video souborů a třída `TImageSource`. Ta dovoluje zacházet s obrázkem stejně, jako by to byl video soubor.

Obě tyto třídy vytvářejí potomka abstraktní třídy `GetFrameThreaded`, který slouží k získávání snímků daného multimediálního souboru, přičemž načítání probíhá v samostatném vlákně. Důvodem tohoto přístupu je snaha oddělit náročné výpočty od vlákna s uživatelským rozhraním, které by tak bylo zbytečně zatíženo a nemuselo by dostatečně rychle reagovat na podněty od uživatele. Načtený snímek je následně uložen do vyrovnávací paměti, která má prostor pro 100

snímků a je sdílena všemi sekvencemi daného multimediálního souboru. Pokud je maximální počet snímků překročen, tak dochází k invalidaci vyrovnávací paměti smazáním nejstaršího uloženého snímku.

Samotné přehrávání multimediálního souboru je zajištěno pomocí časovače, který tiká s periodou nastavenou podle počtu snímků za sekundu daného multimédia. Při vypršení časového limitu dojde k ověření, zda vyrovnávací paměť obsahuje daný snímek. V kladném případě je snímek emitován pomocí signálu pro další zpracování, v opačném případě se čeká, dokud nebude načten. O načítání se stará třída `CachingThread`, která v předstihu zjišťuje, zda vyrovnávací paměť obsahuje daný snímek a v záporném případě dává pokyn druhému vláknu, aby tento snímek načetl. Výhoda tohoto přístupu spočívá v tom, že pokud dojde ke zpoždění během dekódování obrazu, tak se nečeká na vypršení časového intervalu pro žádost o další snímek. Místo toho je následující snímek načten, co nejrychleji je to možné, což v pozitivním směru ovlivňuje plynulost videa.

Renderování videa využívá již zavedené infrastruktury pro přehrávání videa s tím rozdílem, že časovač tiká s co nejmenším zpožděním (1 milisekunda a méně) a výstupní obraz je směřován signálem na slot renderovací jednotky. Časovač bylo nutno použít, jelikož nelze vytvořit renderovací cyklus kvůli asynchronní komunikaci pomocí signálů a slotů mezi jádrem video editoru a komponentou časové osy. Více informací o komponentě časové osy se nachází v kapitole 4.9.

4.4 Komponenta pro zobrazování videa

Tato komponenta je implementována ve třídě `GLMonitor` a slouží pro zobrazování a kopírování náhledu videa nebo obrázků. Třída je potomkem třídy `QGLWidget`, která zapouzdřuje knihovnu OpenGL v Qt Toolkitu. Použití této knihovny jsme si zvolili z důvodu integrované hardwarové akcelerace při vykreslování obrazu.

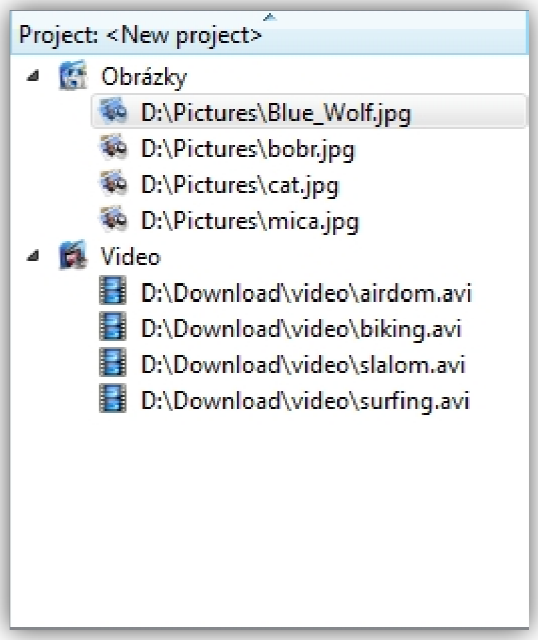
Komponenta používá dvě vyrovnávací paměti pro vykreslování obrazu, které používají barevný model RGB s přidanou informací o alfa kanálu. Obraz, který vykreslujeme, se nachází uvnitř framebufferu, který má šířku a výšku stejnou jako aktuální velikost komponenty. Pokud nemá uživatel vybranou aktivní sekvenci v časové ose nebo nepracuje s dočasnou pracovní kopií, tak je obsah celého framebufferu nastaven na nulu, čímž se v době neaktivity vykresluje černé pozadí. Při změně velikosti monitoru dochází k automatické změně velikosti framebufferu. Velikost uloženého obrazu je změněna metodou nejbližší soused z originálního načteného snímku. Pokud by byla velikost měněna z dat uložených ve framebufferu, tak by postupně docházelo ke stále větší degradaci obrazu.

Metoda nejbližší soused je i přes výslednou kvalitu generovaného snímku výhodná, jelikož je výpočetně rychlá a zubaté hrany, které vzniknout při zvětšení obrazu nejsou při pohybu příliš vidět. Druhá vyzkoušená metoda byla lineární interpolace. Ta však značně ovlivňovala plynulost videa.

Komponenta mimo zobrazování obrazu implementuje i mechanismus drag and drop, který se používá pro vytváření pracovní kopie a kopírování zobrazované sekvence na časové stopy. Jelikož je tento mechanismus obsažen i v komponentě pro správu projektu a časových stopách, bude popsán až detailněji u těchto komponent.

4.5 Komponenta pro správu projektu

Tato komponenta je implementována ve třídě `FileViewWidget` a jejím cílem je poskytnout uživateli přehled o otevřených zdrojových souborech a umožnit mu jednoduché přetahování zdrojů do časových stop a monitoru. Vzhled widgetu je k nahlédnutí na obrázku 4.6.



Obrázek 4.6: Komponenta pro správu projektu s aktuálně otevřenými soubory

Implementace vzhledu a většiny funkcionality je obstarána třídou `QTreeWidget`, která je děděna a upravena pro potřeby Video Editoru. Třída je založena na stromové model/view architektuře, kterou používá Qt Toolkit. Tato architektura dovoluje vytvářet hierarchický strom z položek, které oddělují svojí grafickou reprezentaci od vnitřní datové reprezentace.

Každý otevřený soubor je reprezentován jednou položkou, která je uložena do příslušné multimediální složky. Položka vždy obsahuje popisek v podobě cesty k souboru a ikonu, která uživateli naznačuje typ multimédia. Aby komponenta vypadala nativně pod různými operačními systémy, tak se využívá podmíněného překladu, který formátuje popisek s cestou k souboru dle cílové platformy. Datová část položky je uložena v bytovém poli a obsahuje informace o typu souboru, jeho délce, ukazateli na instanci `TDeviceInterface` a dalších datech. Komponenta je navržena tak, aby se dala snadno rozšiřovat pro další typy multimediálních souborů. Stačí, aby byly dopsány potřebné metody, které vytvoří a naplní položku daty a následně ji umístí do příslušné složky.

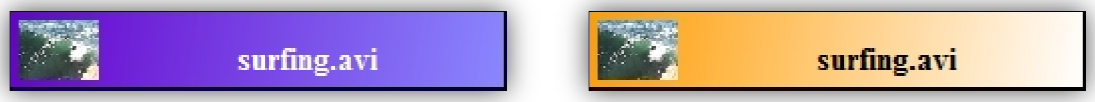
Pro odstranění položky z komponenty ji stačí vybrat a stisknout klávesu Delete. Položka se odstraní a zároveň dojde k emitaci signálu, který informuje další widgety o tom, že mají smazat patřičné prvky pracující se souborem. Když dojde ke zmenšení počtu položek ve složce na hodnotu nula, tak dochází i k automatickému smazání složky.

Pro umisťování zdrojů na časovou osu nebo do komponenty monitoru je implementován mechanismus drag and drop. Při stisknutí tlačítka myši dojde u uložení počáteční pozice kurzoru a při tažení myší se počítá délka, kterou myš urazila. Operace přetažení je spuštěna až v okamžiku, kdy myš přesáhne minimální vzdálenost, která je definována v patřičné konstantě Qt Toolkitu. Výhoda tohoto přístupu je, že nedochází k nechtěné změně chování komponenty v případě, že se uživateli jen mírně třese ruka při označování jednotlivých položek.

Po přesáhnutí minimální délky se vytvoří objekt třídy `QMimeData`, který slouží jako kontejner pro přetahovaná data a zároveň popisuje jaký typ dat je v kontejneru uložen. Popiska pro data je dle standardu kódování MIME. Jelikož jsou naše data uložena ve vlastním bytovém formátu, zvolili jsme neoficiální popisku x application/sourcedata. Po uložení všech potřebných dat z datové části aktuálně vybrané položky je vytvořen objekt třídy `QDrag`, který zapouzdřuje celou drag and drop operaci. Po jejím dokončení je tento objekt Qt Toolkitem automaticky smazán.

4.6 Komponenta pro reprezentaci sekvence

Tato komponenta je implementována ve třídě `TimelineObject` a slouží k zobrazení grafické reprezentace sekvence, která je umístěna na časové ose. Hlavní důraz při návrhu komponenty byl kladen na to, aby komponenta uživateli poskytla rychlý přehled o tom, jaké sekvence jsou umístěny na časové ose. Mimo jiné by komponenta měla zobrazovat náhled daného multimédia v případě, že je to jen trochu možné. Vzhled komponenty je zobrazen na obrázku 4.7.



Obrázek 4.7: Komponenta pro reprezentaci sekvence - vlevo neaktivní stav, vpravo aktivní stav

Při vytvoření komponenty dochází k automatickému vytvoření objektu `TDeviceInstance`, který reprezentuje sekvenci multimediálního souboru. Veškerou správu nad tímto objektem pak přebírá tato grafická komponenta.

Vzhled widgetu je dán jeho šířkou a aktivitou. Aktivní a neaktivní widget je zobrazen na obrázku 4.7. Aktivní komponenta se rozpoznává pomocí statické proměnné, která je sdílena všemi instancemi třídy `TimelineObject`. Pokud má komponenta dostatek prostoru, vykresluje se vlevo první snímek sekvence a vpravo text nesoucí informaci o souboru. V opačném případě dochází k vypuštění některých součástí vzhledu. Priorita, která byla zvolena při redukování vykreslovaných prvků, se zaměřuje na vykreslení náhledu multimédia před jménem souboru. Pokud je však komponenta příliš malá, aby se mohl náhled vykreslit, je vhodné, aby se zobrazila alespoň část názvu souboru. To je dokumentováno na obrázku 4.8.

Obrázek 4.8: Vlevo střední velikost a priorita náhledu, vpravo nejmenší velikost a priorita textu

4.7 Komponenta časové stopy

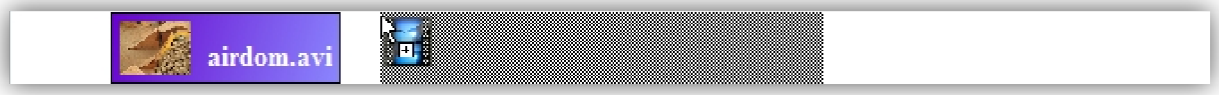
Tato komponenta je implementována ve třídě `TimelineTrack` a jejím hlavním účelem je poskytovat kontejner pro objekty třídy `TimelineObject`. Mimo jiné by měla být tato komponenta uživatelsky přívětivá, čili od ní očekáváme, že bude při vkládání nové sekvence zobrazovat náhled její velikosti. Dále by měl widget umět přibližovat a oddalovat umístěné sekvence, což je výhodné v případě, že je sekvence příliš krátká nebo příliš dlouhá a manipulace s ní by byla pro uživatele obtížná. Vzhled komponenty je vidět na obrázku 4.9.

Obrázek 4.9: Vzhled komponenty časové stopy

Umístění sekvence do kontejneru probíhá pomocí mechanismu drag and drop z okna pro správu projektu nebo z monitoru. Při přetažení se zjišťuje typ dat pomocí MIME popisky. Pokud

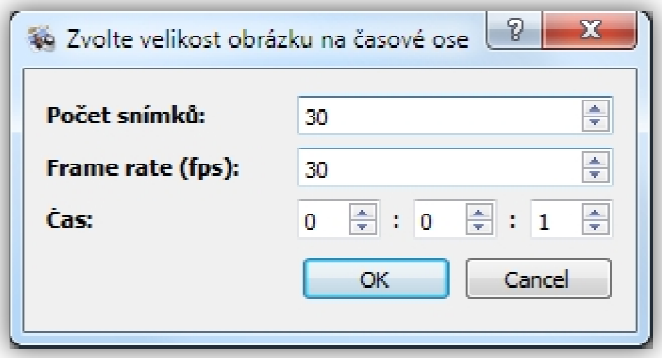
popiska obsahuje náš neoficiální typ x application/sourcedata, je povolena možnost upustit data do kontejneru. Mimo jiné dojde během přetahování k nastavení potřebných vnitřních proměnných a k zavolání překreslení komponenty pro zobrazení náhledu velikosti vkládané sekvence.

Náhled se vykresluje jako šedý obdélník s hustým dlaždicovým vzorem. Šířka obdélníku závisí na délce sekvence v časové jednotce vynásobena počtem pixelů na 1 sekundu a ovlivněna aktuálním přiblížením nebo oddálením časové stopy. Výsledek je k nahlédnutí na přiloženém obrázku 4.10.



Obrázek 4.10: Náhled délky vkládané sekvence

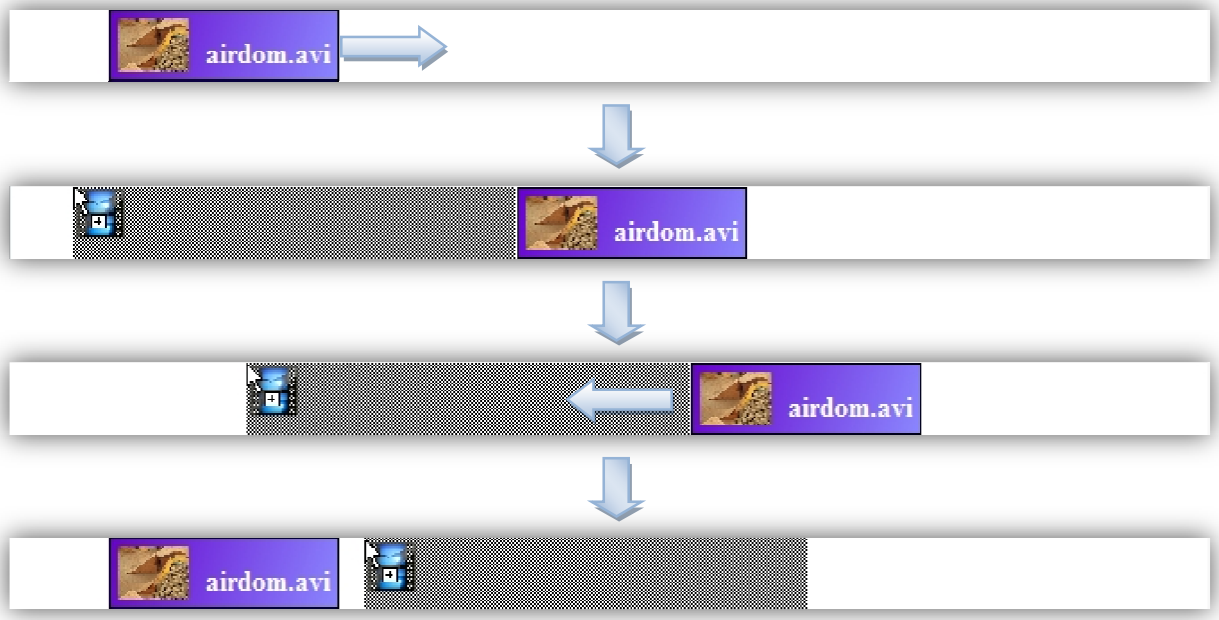
Při uvolnění levého tlačítka myši dojde k dekódování veškerých dat přetahované sekvence, které jsou uloženy v datové části objektu `QDrag`. Následně je dle typu sekvence rozhodnuto, jakou akcí se bude pokračovat. V případě, že se jedná o sekvenci s videem, dochází k okamžitému vytvoření instance třídy `TimelineObject`, které se předají potřebné data. V případě, že se jedná o sekvenci vytvořenou ze statického obrázku, dochází před vytvořením instance k zobrazení dialogového okna, které nechá uživatele nastavit počáteční parametry. Po potvrzení zadaných údajů se pokračuje stejným způsobem jako u sekvence videa. Na obrázku 4.11 jsou vidět veškeré parametry, které může uživatel při vkládání statického obrazu navolit. Při změně jakéhokoliv parametru jsou ostatní parametry automaticky přepočítány.



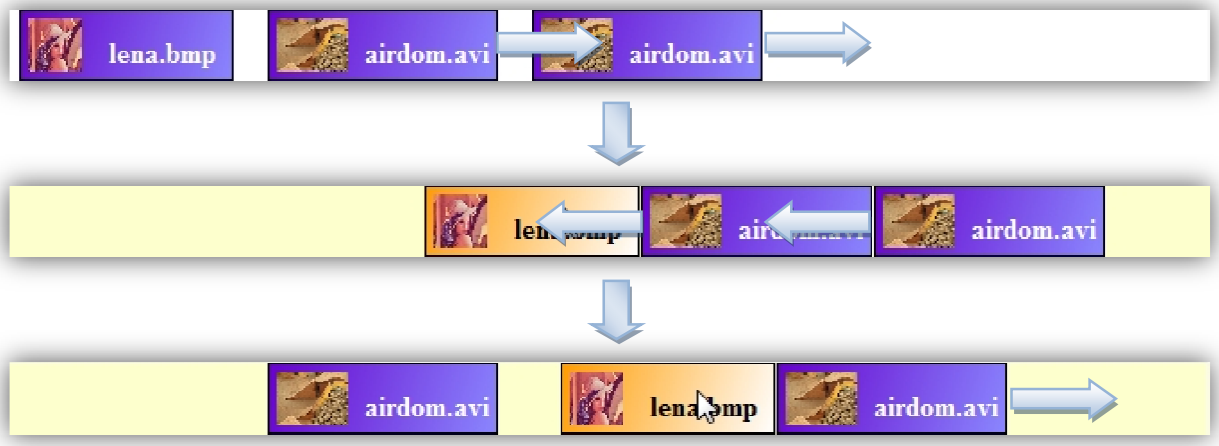
Obrázek 4.11: Nastavení počátečních vlastností obrázku

Po vytvoření komponenty pro grafickou reprezentaci sekvence dochází k jejímu umístění do kontejneru, který přebírá veškerou správu nad tímto objektem. Kontejner automaticky určuje a přepočítává geometrii tohoto widgetu v závislosti na úpravách sekvence. Veškeré informace o umístění komponent na časové stopě najdeme v uspořádaném seznamu struktur `TWidgetPos`.

Animační framework je novinka Qt Tookitu v aktuální verzi 4.6.1. Framework poskytuje třídu `QPropertyAnimation`, která slouží k animaci pomocí atributů widgetu. Třídě se předává ukazatel na widget, který se má animovat, dále se nastaví, jaký atribut se bude animovat a jak dlouho má animace trvat. Doba trvání animace komponenty pro reprezentaci sekvence trvá 1 sekundu a ovlivňovaná vlastnost je geometrie objektu. Spuštění animace nastává v případě, že vkládaná sekvence koliduje s jinou již vloženou sekvencí na časové stopě. Animace zajistí, aby došlo ke kaskádovitému posunutí všech komponent pro reprezentaci sekvence napravo od vkládané sekvence. Stejný přístup je uplatněn i v případě, kdy se s uloženou sekvencí na časové ose hýbe. Celý proces je demonstrován na přiložených obrázcích 4.12 a 4.13.



Obrázek 4.12: Průběh animace při vkládání nového souboru



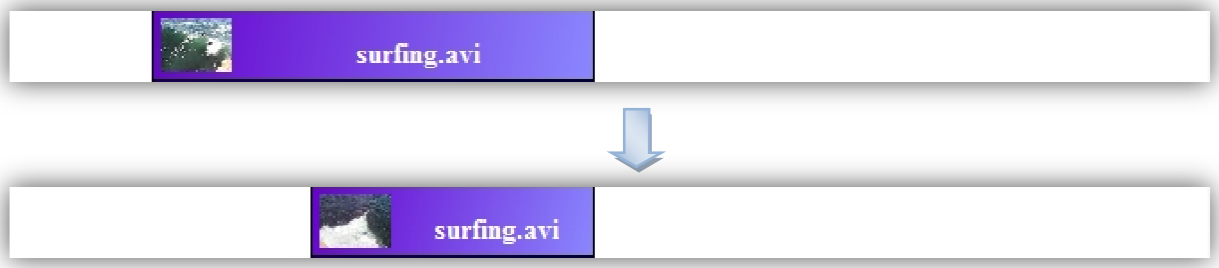
Obrázek 4.13: Kaskádovitý průběh animace při posunování vloženého souboru

Po vložení nové sekvence do časové stopy nebo dokončení přetažení již vložené sekvence dojde k uložení nových pozic všech posunutých komponent. Nakonec je seznam s veškerými změněnými údaji setříděn metodou Quick Sort, aby byla zachována platnost podmínek pro další možné posouvání. Při posunutí widgetu dochází ještě k následujícím jevům. Prvním je automatické zvětšení šířky stopy v případě, že pravý okraj přesouvaného nebo posunutého widgetu přesáhne aktuální šířku kontejneru. Druhým jevem je emitace signálu, která nese informaci o tom, s kterým objektem se právě pohnulo. Obě tyto vlastnosti jsou využívány v komponentě časové osy, která je probírána v kapitole 4.9.

Každá stopa má implementován vykreslovací mechanismus, který mimo náhledu právě vkládané sekvence vykresluje i aktivitu stopy pomocí světle žluté barvy. Aktivita je podobně jako u komponenty pro grafickou reprezentaci sekvence řešena pomocí statické proměnné.

Časová stopa se taktéž stará o změnu geometrie vložených komponent v případě oříznutí sekvence zleva nebo zprava. Způsob změny geometrie je ilustrován na obrázku 4.14. V případě, že dojde

k oříznutí sekvence zprava, dochází ke změně geometrie vloženého objektu obdobným způsobem z druhé strany.



Obrázek 4.14: Změna pozice při oříznutí zleva

Každou vloženou komponentu v kontejneru lze smazat. Pro smazání stačí komponentu vybrat a stisknout tlačítko Delete. Mimo smazání je možnost komponenty přiblížit nebo oddálit. To lze provést pomocí kláves plus a mínus, případně pomocí kolečka myši.

4.8 Komponenta časového pravítka

Tato komponenta je implementována ve třídě `TimelineTimeWidget` a jejím hlavním cílem je poskytnout uživateli přehled o časovém úseku, s kterým aktuálně pracuje. Komponenta musí zvládat změnu časové měřítka v případě, že dojde k přiblížení nebo oddálení časových stop. Widget by měl navíc disponovat jezdcem, s kterým může uživatel manipulovat a ovládat pomocí něj aktuální pozici přehrávací hlavy na časové ose. Na obrázku 4.15 je vzhled této komponenty.

Obrázek 4.15: Vzhled komponenty pro časovou osu s přiblížením 1.00x

Největší část funkcionality této komponenty je implementována v metodě pro vykreslování, která po celé délce widgetu kreslí ve správném rozestupu dílky časového pravítka. Jedna sekunda je v přiblížení 1.00x dlouhá 30 pixelů. Tato hodnota byla zvolena empirickým způsobem z velikosti sekvence umístěné na časové ose. Menší hodnota způsobovala, že sekvence byly příliš malá a špatně se s ní pracovalo. Větší hodnota naopak způsobovala značnou nepřehlednost při umístění delších sekvencí.

Mezi jednotlivými časovými údaji jsou vykresleny rovnoměrně mezidílky, které slouží k lepšímu odhadu vzdálenosti. Pokud dochází k oddálení stop, tak se automaticky oddaluje i časové pravítko a redukuje se text s informací o čase, tak aby tato komponenta byla vždy přehledná. Na obrázcích 4.16 a 4.17 je vidět časové pravítko ve stavu největšího podporovaného přiblížení a největšího oddálení.

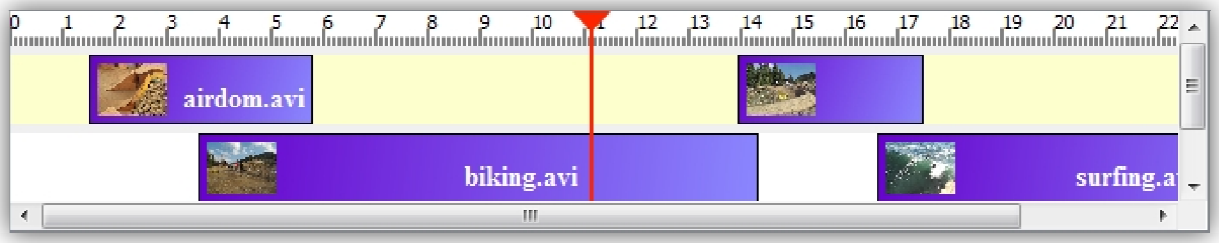
Obrázek 4.16: Časové pravítko s přiblížením +5.00x

Obrázek 4.17: Časové pravítko s oddálením -5.00x

Jezdec pro přehrávací hlavu je vykreslen jako polygon složený ze tří bodů s červenou výplní. Z vrcholu jezdce vychází čára, která sahá až ke spodnímu okraji widgetu. Jezdce je možné přesunout jeho uchopením a táhnutím na požadovaný čas nebo pouhým klepnutím na dané místo. Výsledná cílová pozice jezdce v časové jednotce je spočítána z pozice kurzoru myši a aktuálního přiblížení.

4.9 Komponenta časové osy

Komponenta časové osy zapouzdřuje funkcionalitu časových stop a časového pravítka. Jedním z hlavních cílů, které si komponenta klade, je poskytnutí možnosti přidávat, odstraňovat a přesouvat časové stopy. V případě, že je okno s časovou osou příliš malé a časové stopy zabírají více prostoru než je aktuální velikost této komponenty, tak by mělo dojít k automatickému zobrazení posuvníků, které uživateli nabídnou možnost zobrazit zbývající vnořené widgety. V poslední řadě by se měla komponenta starat o přehrávání náhledu multimédií ze všech časových stop ve správném pořadí. Implementace tohoto widgetu se nachází ve třídě `TimelineWidget` a výsledný vzhled časové osy je k nahlédnutí na obrázku číslo 4.18.



Obrázek 4.18: Vzhled komponenty časové osy

Po inicializaci obsahuje komponenta časové pravítko a tři časové stopy, s kterými může uživatel v prvopočátku pracovat. Geometrie všech vytvořených objektů je nastavena na šířku komponenty časové osy. Mezi jednotlivými vnořenými widgety je vždy mezera široká 5 pixelů, která má estetický a zároveň i praktický význam. Praktický význam spočívá v přehlednosti jednotlivých stop, které by kvůli svému bílému podkladu bez této mezery splynuly v jednolitou bílou plochu, na které by se uživatel špatně orientoval. Všechny stopy se ukládají do uspořádaného seznamu, s kterým jsou prováděny veškeré operace.

Přidávání a mazání stop je vyvoláno patřičným tlačítkem v hlavním okně Video Editoru. Nová stopa je přidána pod právě aktivní stopu. V případě mazání stopy je aktivní stopa odstraněna a aktivita je předána další nejbližší vyšší nebo nižší stopě.

Posouvání stop se děje po zmáčknutím příslušného tlačítka v hlavním okně. Obslužná metoda v komponentě časové osy prohodí posouvanou stopu se sousední stopou uvnitř uspořádaného seznamu a následně dojde k nastavení nové pozice prohozených stop uvnitř komponenty časové osy. Při operaci posouvání jsou emitovány dva signály. První informuje hlavní okno o tom, zda by mělo zablokovat tlačítko pro posouvání stopy v případě, že jsme stopu umístili na počáteční nebo koncovou pozici. Druhý signál nese informaci o tom, s kterou stopou bylo pohnuto. Tato informace je využívána objektem třídy `QScrollArea`, který se stará o zobrazování posuvníků. Informace o posouvaném widgetu slouží k vycentrování aktuálního výřezu tak, aby byla komponenta vždy viditelná během posouvání. Stejný princip se využívá i při posunutí sekvence na časové stopě mimo aktuální výřez.

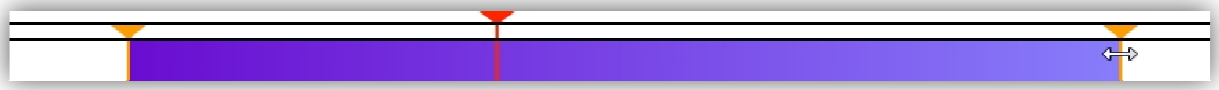
Objekt třídy `QScrollArea` zobrazuje posuvníky automaticky, když dojde ke změně velikosti komponenty časové osy. Ke změně výšky widgetu dochází přidáním nebo odebráním časových stop. Změna šířky je ovlivněna šířkou časových stop. V případě, že dojde k automatickému roztáhnutí časové stopy vlivem přetažení sekvence za nejpravější okraj, tak se časová osa dynamicky přizpůsobí a zároveň upraví i ostatní vnořené widgety na správnou velikost.

Na obrázku 4.18 je vidět ryska přehrávací hlavy, která zdánlivě patří k časovému pravítku a sahá až k spodnímu okraji komponenty časové osy. Tohoto efektu je docíleno pomocí vnořeného widgetu, který vykresluje tuto čáru. Tento způsob byl zvolen, jelikož vykreslení je velmi rychlé a jednoduché. Navíc je možné rysku velice snadno zvětšit nebo zmenšit při přidávání a odebírání stop. Druhý způsob, který dociluje tohoto efektu, by byla distribuce pozice přehrávací hlavy z časového pravítka do zbylých komponent, které by čáru vykreslily. Tento způsob byl také vyzkoušen, ale nakonec byl zavrhnut, jelikož velice zpomaloval přehrávání neustálým překreslováním velkého počtu komponent.

Přehrávání náhledu celého scénáře je řešeno následujícím způsobem. Nejdříve proběhne vyhledávání, které se dotazuje jednotlivých stop, zda v daném čase obsahují nějakou sekvenci. Pokud ano, tak je vyhledáváno přerušeno. V opačném případě je spuštěn vnitřní časovač, který tiká v intervalu $\frac{1}{30}$ sekundy a posunuje přehrávací hlavu. To je výhodné v případě, kdy uživatel neumístí sekvence přesně za sebe nebo přes sebe na rozdílných stopách. Po posunutí času vnitřním časovačem se vyhledávání znovu opakuje. Při nalezení sekvence na nějaké časové stopě dojde k ověření, zda se již jiná sekvence nepřehrává nebo neběží námi spuštěný časovač. Pokud ano, dojde k jejich zastavení. Následně se spustí přehrávání nové sekvence od správného času a cyklus se po přehrání snímku sekvence začne opakovat. Přehrávání končí v případě, kdy přehrávací hlava dorazí na konec časové osy nebo v případě, kdy uživatel přehrávání přeruší. Procházení časových stop při renderování výsledného videa je implementováno obdobným způsobem, ale interval posouvání hlavy je nastaven na nejkratší možný interval.

4.10 Komponenta pro ořezávání sekvencí

Pro ořezání sekvence videa s přesností na 1 snímek byla vyvinuta komponenta, která má 2 táhla určující počáteční a koncový bod střihu. Widget navíc poskytuje jezdce, který slouží k nastavení přehrávací hlavy náhledu ořezávaného videa. Implementace lze nelézt ve třídě `SpanSlider`. Vzhled komponenty je vidět na obrázku 4.19.



Obrázek 4.19: Vzhled komponenty pro ořezávání videa

Jezdci na obou okrajích určují počáteční a koncový bod střihu. Hodnoty jezdců se pohybují v intervalu <1; celkový počet snímků>. Jezdec přehrávací hlavy se pohybuje mezi počátečním a koncovým bodem střihu a jeho hodnota je v intervalu <0; celkový počet snímků – 1>. Důvodem je zachování podmínky při přehrávání, kdy aktuální snímek musí být menší než koncový bod střihu.

Krajní jezdci pro ořezávání videa se nastavují uchopením a tažením myši na požadovanou pozici. Aktivní oblast je definována jako obdélník, který ohraničuje jezdce. V rámci uživatelské přívětivosti je aktivní oblast uživateli znázorněna změnou kurzoru myši. S jezdcem přehrávací hlavy se dá

pohybovat stejným způsobem jako s jezdci pro ořezávání videa. Navíc je k dispozici i druhý způsob, který přesune přehrávací při kliknutí kamkoliv do prostoru ořezávané sekvence. Pozice přehrávací hlavy se vždy pohybuje mezi počátečním a koncovým bodem střihu. To je dáno implementací jádra, které nedovoluje přehrávat snímky, které se nachází před nebo za body střihu.

5 Omezení a rozšíření

Program Video Editor má určitá omezení, která vyplynula z použitých implementačních prostředků a návrhu aplikace. Tato kapitola na tyto omezení upozorňuje a navrhuje možné způsoby jejich odstranění.

5.1 Vstup a výstup

Omezení pro vstup a výstup vyplývají z použité knihovny OpenCV. Statický obraz je možný načíst ve formátech **BMP**, **DIB**, **JPG**, **JPEG**, **PNG** a **TIFF**. Video soubory musejí být uloženy v kontejneru AVI v kombinaci s kodeky **RGB(A) ('DIB')**, **RAW I420 ('I420')**, **RAW I420 ('IYUV')**, přičemž každý snímek videa musí být klíčový. Komprimované video, které používá ztrátové kodeky jako je DivX, Xvid a další, nelze korektně načíst. Výstupní video je uloženo v kontejneru AVI pomocí kodeku MPEG1. Rozlišení a počet snímků za sekundu cílového videa si uživatel může zvolit dle potřeby sám. Knihovna OpenCV neumí pracovat se zvukovou stopou videa, takže uživatel nemá možnost ozvučit výstupní video dle své potřeby.

Vhodnou alternativou k OpenCV by byla knihovna FFMpeg [10], která obsahuje sadu kodérů a dekodérů pro práci s audiem a videem včetně komprimovaných formátů. Tato knihovna nebyla v počátku zvolena, jelikož její rozhraní je značně složité a práce s ní by zabrala značnou část času pro vypracování práce. Bylo by však do budoucna vhodné, aby tato knihovna nahradila stávající knihovnu OpenCV a to hlavně z důvodu, že OpenCV načítá automaticky všechny snímky v RGB formátu. Následně musíme daný snímek znovu načítat z paměti a transformovat ho do RGBA formátu, což značně zpomaluje vykreslování videa.

5.2 Efekty

Aplikace neumí nanášet efekty na jednotlivé sekvence. Vhodným řešením by bylo přidání efektového zásobníku ke komponentě pro grafickou reprezentaci sekvence. Tento zásobník by se mohl plnit efekty buď z projektového okna pomocí mechanismu drag and drop nebo vyvoláním příslušného dialogového okna pomocí kontextového menu nad cílovou sekvencí. Pro přechody mezi jednotlivými sekvencemi by mohla sloužit samotná komponenta pro grafickou reprezentaci sekvence, která se umístí do stopy nad sousední sekvence a nastaví se jí příznak, že jde o přechod.

Nanášení efektů na jednotlivé snímky by mohlo být zpracováno pomocí komponenty, která se připojí na výstupní signál z jádra pro zpracování videa. Tato komponenta následně obraz upraví a přepošle výsledek na vstupní slot monitoru nebo renderovací jednotky. Jednotlivé efekty by mohly být řešeny jako dynamické knihovny, které se umístí do složky spolu se spustitelným souborem aplikace. Tyto knihovny by byly za běhu aplikace automaticky načteny a umožňovali by dané efekty používat. Tento přístup navíc umožňuje snadné rozšiřování funkcionality aplikace i třetími stranami, kterým stačí jen definované rozhraní nebo abstraktní třída pro vývoj nových efektů. Informace o vytváření zásuvných modulů v prostředí Qt Toolkitu poskytuje kapitola Making Applications Plugin Aware v knize [7].

6 Závěr

Cílem této práce bylo navrhnout a vytvořit jednoduchý program pro střih videa, který by měl intuitivní uživatelské rozhraní. Při návrhu aplikace se vyházelo ze zkušeností s obdobnými programy, u kterých byly identifikovány společné rysy, které pak byly zaneseny i do vytvářené aplikace. Důraz při implementaci uživatelské rozhraní byl kladen na intuitivnost. Tento cíl se podařilo splnit, jelikož skupina testujících uživatelů byla schopna program efektivně používat během prvních 10 minut po jeho zapnutí.

Od této testovací skupiny vzniklo i spousta návrhů a připomínek pro další vývoj aplikace. Uživatelé by nejvíce ocenili, kdyby aplikace uměla pracovat s audiem a umožňovala přidávat efekty k jednotlivým sekvencím. Mezi nejvíce žádané efekty bylo přidání titulků do videa a plynulé prolnutí dvou sekvencí. Dalším nejčastějším požadavkem bylo zpracování komprimovaného videa. Pro implementaci těchto požadavků by bylo potřeba použít jinou knihovnu v jádře, která by dokázala načítat audio i video. Jako nejvhodnější varianta se jeví open source knihovna FFMpeg. Pro snadnou implementaci všech rozšíření vznikla programátorská dokumentace, která obsahuje zároveň návod pro instalaci všech potřebných knihoven. Dokumentace je součástí přiloženého DVD. Celý obsah DVD je uveden v sekci Přílohy.

Vlastní přínos práce pro mou osobu spočíval v tom, že jsem hlouběji porozuměl práci s video soubory a dále jsem se seznámil s tvorbou uživatelských komponent pro grafické uživatelské rozhraní. Tato znalost se mi bude hodit do budoucna, až budu vytvářet nějaký další program, který by měl mít specifické ovládací prvky. Jako další přínos bych viděl hlubší porozumění knihovny Qt Toolkit než jsem měl doposud, což se také bude hodit, jelikož společnost Nokia plánuje prosadit tuto knihovnu do svých chytrých mobilních zařízení. Tím by měla umožnit vývojářům psát aplikace pro mobilní telefony jednoduše pomocí C++ místo obvyklé Javy.

Literatura

[1] Adobe Premiere Pro In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 28 October 2005, 16 April 2010 [cit. 2010-04-26]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Adobe_Premiere_Pro>.

[2] *Kdenlive* [online]. 26th February 2010 [cit. 2010-04-26]. Dostupné z WWW: <http://www.kdenlive.org/>.

[3] *Co je Kdenlive* [online]. 29.11.2009 [cit. 2010-04-26]. Wikiknihy. Dostupné z WWW: <http://cs.wikibooks.org/wiki/Kdenlive/Co_je_Kdenlive>.

[4] *Kdenlive* [online]. 18.9.2006 [cit. 2010-04-26]. ABC Linuxu. Dostupné z WWW: <http://www.abclinuxu.cz/software/multimedia/video/editory/kdenlive>.

[5] *OpenShot Video Editor* [online]. c2008 [cit. 2010-04-26]. Dostupné z WWW: <http://www.openshotvideo.com/>.

[6] *A cross-platform application and UI framework* [online]. 2010-04-07 [cit. 2010-04-07]. Qt. Dostupné z WWW: <http://qt.nokia.com/>.

[7] BLANCHETTE, Jasmin; SUMMERFIELD, Mark. *C++ GUI Programming with Qt 4* [online]. Second Edition. New Jersey : Prentice Hall, February 04, 2008 [cit. 2010-04-07]. Making Connections, s. . Dostupné z WWW: <http://www.amazon.com/C-GUI-Programming-Qt4-ebook/dp/B0013TX6YE>. ISBN 0-13-714397-4, 978-0-13-714397-9.

[8] BLANCHETTE, Jasmin; SUMMERFIELD, Mark. *C++ GUI Programming with Qt 4* [online]. Second Edition. New Jersey : Prentice Hall, February 04, 2008 [cit. 2010-04-07]. Subclassing QDialog, s. . Dostupné z WWW: <http://www.amazon.com/C-GUI-Programming-Qt4-ebook/dp/B0013TX6YE>. ISBN 0-13-714397-4, 978-0-13-714397-9.

[9] *OpenCV Wiki* [online]. 2010-04-07 [cit. 2010-04-07]. Welcome. Dostupné z WWW: <http://opencv.willowgarage.com/wiki/>.

[10] *FFMpeg* [online]. 2010-04-14 [cit. 2010-04-14]. Dostupné z WWW: <http://www.ffmpeg.org/>.

Přílohy

Přiložené DVD obsahuje:

- Zdrojový kód programu (videoeditor/src/)
- Programátorskou dokumentaci (videoeditor/doc/index.html)
- Testovací data (videoeditor/video/ a videoeditor/image/)
- Ukázky výstupních video sekvencí (videoeditor/examples/)
- Text práce v elektronické podobě (text/editorvidea.pdf)
- Prezentační plakát (text/plakat.pdf)