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

VIDEO EDITOR SPORTOVNÍCH ZÁZNAMŮ

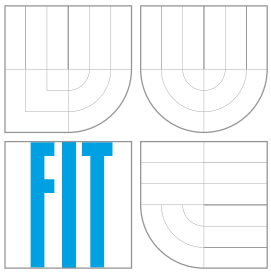
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN JANOUŠEK

BRNO 2015

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

VIDEO EDITOR SPORTOVNÍCH ZÁZNAMŮ

VIDEO EDITOR OF SPORT RECORDINGS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN JANOUŠEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ALEŠ LÁNÍK

BRNO 2015

Abstrakt

Tato práce se zabývá tvorbou video editoru pro mobilní zařízení s operačním systémem Android. Je zde představen operační systém Android a další použité technologie, včetně knihovny Ing. Tomáše Slavotínka, která je využita pro zpracování videa. Aplikace rovněž umí zpracovávat informace z GPS a jiných senzorů. Výsledné projekty jsou automaticky ukládány do databáze SQLite a je možnost je exportovat do video souborů.

Abstract

This thesis deals with creating of video editor for mobile devices running on Android operating system. It is introduced Android operating system and other technology used, including the Ing. Tomas Slavotinek's library, which is used for video processing. The application is also able to process data from GPS and other sensors. The output projects are stored in SQLite database automatically and is the ability to export them to video files.

Klíčová slova

video editor, android, sportovní záznamy, ORMLite, senzory

Keywords

video editor, android, sport recordings, ORMLite, sensors

Citace

Martin Janoušek: Video editor sportovních záznamů, bakalářská práce, Brno, FIT VUT v Brně, 2015

Video editor sportovních záznamů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Aleše Láníka.

.....
Martin Janoušek
17. května 2015

Poděkování

Zde bych chtěl poděkovat vedoucímu práce Ing. Aleši Láníkovi za cenné odborné rady a vedení této práce.

Obsah

Kapitola 1

Úvod

Pořizování sportovních videozáznamů je čím dál více populárnější. K tomuto rozvoji přispěly zejména kamery GoPro, které především díky své minimální velikosti a vysoké odolnosti získaly na obrovské popularitě. Dnes se na trhu samozřejmě objevuje několik dalších výrobců podobných zařízení a počty prodaných kamer, a tím i pořízených záznamů, velmi rychle stoupá.

Stejně tak jako vzrostl počet majitelů těchto sportovních kamer, vzrostl počet uživatelů chytrých telefonů, které již v současné době nabízí dostatečný výpočetní výkon na zpracování těchto videí. Dříve bylo nutné videa zpracovávat na počítačích, přičemž řada lidí ani nevlastnila nástroje pro jejich úpravu. Další nevýhodou byla nutnost strávit určitý čas při úpravě videa přímo u počítače. Oproti tomu úprava videa na mobilním zařízení může být prováděna prakticky kdekoli, a využívat tak k tomu například čas při cestě autobusem, nebo při čekání u lékaře.

Cílem této práce je seznámit se s mobilní platformou Android a dále navrhnout a implementovat video editor pro tuto mobilní platformu. Tento video editor by měl být přizpůsobený přímo pro střih sportovních videí a dále by měl umět zpracovat data nasbíraná při jejich pořizování a vhodně je interpretovat.

V kapitole 2 je vysvětleno několik základních pojmů, které jsou potřeba k pochopení dalších částí této práce.

Kapitola 3 představuje několik knihoven pro práci s multimediálními soubory, ze kterých je následně vybrána knihovna použitá pro tuto práci.

V kapitole 4 je přehled dosud existujících řešení, které jsou představeny od nejjednodušší aplikace po nejsložitější.

V kapitole 5 a 6 je popsán návrh a implementace samotné aplikace a v posledních dvou kapitolách 7 a 8 jsou shrnuty výsledky včetně závěrečného hodnocení.

Kapitola 2

Android

Operační systém Android[7], dostupný jako open source, je určen zejména pro mobilní zařízení. V dnešní době se ovšem rozrůstá i do dalších elektronických zařízení jako jsou televize, nositelná elektronika nebo palubní počítače automobilů. Android vznikl jako produkt firmy Android Inc., kterou později odkoupil Google Inc. a vytvořil z něj nejpoužívanější mobilní platformu na trhu (březen 2015, zdroj: [13]). Společně s Google Inc. vyvíjí tento systém konsorcium Open Handset Alliance[12], jakožto seskupení výrobců mobilních zařízení, mobilních operátorů, vývojářů softwarových produktů a dalších.

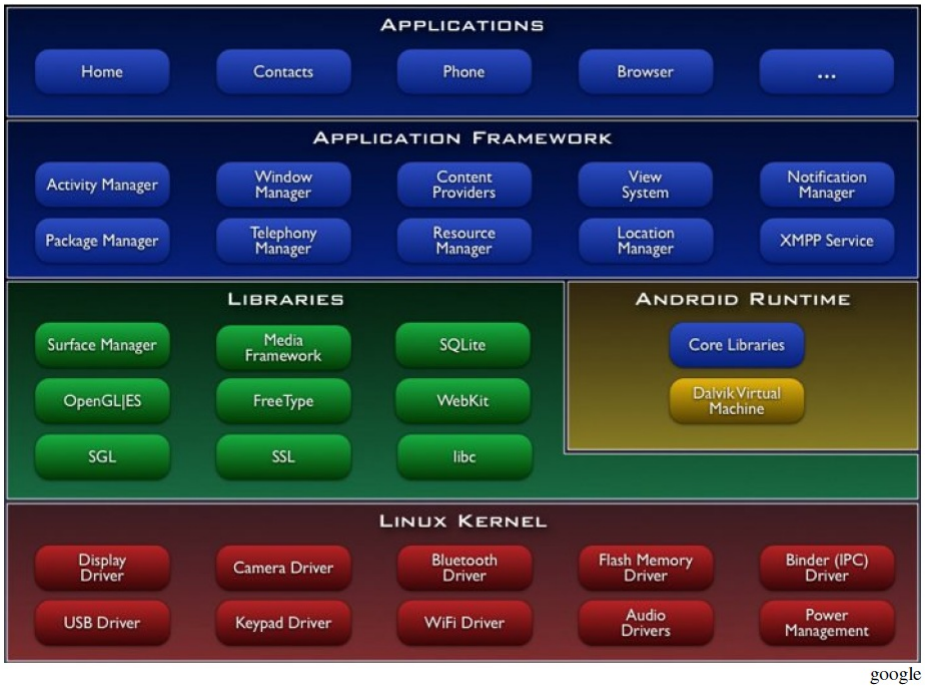
Android je založen na Linuxovém jádře a je přizpůsoben tomu, aby byl schopen fungovat na zařízení s omezenými zdroji (omezená výdrž baterie, relativně malý výpočetní výkon, různá velikost displeje apod.). Logickým důsledkem je proto fakt, že některé prvky tohoto mobilního operačního systému jsou zjednodušeny a odlehčeny[1].

V současnosti je nejnovějším OS Android verze 5.1 s názvem Lollipop[4], která prozatím ještě není využívána většinou uživatelů. V přehledu níže lze vidět využívání jednotlivých verzí OS Android.

Version	**Codename**	**API**	**Distribution**
2.2	Froyo	8	0.4 %
2.3.3 - 2.3.7	Gingerbread	10	6.4 %
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.7 %
4.1.x	Jelly Bean	16	16.5 %
4.2.x		17	18.6 %
4.3		18	5.6 %
4.4	KitKat	19	41.4 %
5.0	Lollipop	20	5.0 %
5.1		21	0.4 %

Tabulka 2.1: Přehled verzí systému OS Android k 6. 4. 2015 [4]

Téma operačního systému Android je velmi obsáhlé, a proto je zde uvedeno jen několik základních informací, které jsou důležité pro pochopení určitých částí této práce.



Obrázek 2.1: Architektura operačního systému Android. [6]

2.1 Architektura systému

Jak bylo uvedeno, Android je založen na linuxovém jádře, které je pochopitelně psané v jazyce C/C++, ale mobilní aplikace, jako je například právě tato, je psaná v jazyce Java. Proto je potřeba vysvětlit, jak jednotlivé části OS Android pracují.

Android je rozdělen do několika vrstev, přičemž každá vrstva, má na starosti určitou funkcionalitu a komunikuje s okolními vrstvami.

- **Linux kernel** – nejnižší vrstva je základ operačního systému, kterou tvoří linuxové jádro. Konkrétně je u nejnovějšího Androidu 5.0 verze jádra 3.14 [19]. Tato vrstva je poslední softwarovou vrstvou, která spojuje operační systém s hardwarem. Proto mimo jiné obsahuje ovladače pro práci s periferními zařízeními[2], ale také zajišťuje správu procesů, paměti, napájení a dalších základních funkcí. S touto vrstvou uživatel přímo nikdy nepracuje, ale její možnosti jsou zprostředkovávány vyššími vrstvami. Jádro je napsané v jazyce C a C++.

- **Libraries** – další vrstvu tvoří nativní knihovny, které poskytují základní funkce systému. Nachází se zde knihovna pro práci s databází SQLite[3], dále grafické knihovny SGL a Open GL (pro práci s 2D a 3D grafikou), knihovna WebKit, která je jádrem pro webové prohlížeče a další. Knihovny jsou psány v jazyce C a C++, stejně jako jádro.

- **Android Runtime** – součástí vstvy nativních knihoven je část nazvaná *Android Runtime*, která zastřešuje Dalvik Virtual Machine (DVM), který se stará o překlad

[1]Mezi zjednodušené prvky systému OS Android patří například SQLite nebo Dalvik Virtual Machine.
[2]Kamera, bluetooth, USB, displej, atd.
[3]Odlehčená verze klasické SQL databáze.

aplikací v jazyce Java. Konkrétně se jedná o kompilaci tzv. Dalvik bytecode a to do kódu určeného přímo pro konkrétní cílovou architekturu. Rozdíl mezi JVM (Java Virtual Machine) a Dalvikem je v optimalizaci na mobilní zařízení. Na rozdíl od JVM totiž Dalvik využívá .dex soubory, které jsou pro mobilní zařízení kompaktnější. Dalvik je založen na kompilaci *just in time*. To znamená, že části kódu jsou překládány vždy ve chvíli kdy jsou vyžadovány[4]. Nově od Androidu verze 4.4 (Kitkat) je možné využít nové technologie překladu ART, která překládá *Dalvik bytecode* do *system-dependent binary*. Celý kód aplikace je tedy předpřeložen během instalace (pouze jednou)[17]. Od verze Android 5.0 byl Dalvik nahrazen právě systémem ART.

- **Application Framevork** – následující vrstva nazvaná Application Framevork je první vrstvou napsanou v jazyce Java a nabízí podporu pro systémové a uživatelské aplikace. Obsahuje knihovny a jejich API pro snadné vytváření aplikací. Příkladem může být Activity Manager (stará se o správu Aktivit), Notification Manager (spravuje veškeré oznámení od aplikací), Content Providers (umožňují sdílení dat mezi aplikacemi), Resource Manager (spravuje přístup k externím zdrojům) a mnohé další, které jsou vysvětleny přímo na stránkách Androidu pro vývojáře [6].

- **Application** – poslední vrstvou, která se ve schématu nachází je vrstva samotných aplikací. V této poslední vrstvě jsou všechny aplikace, které můžeme mít v systému. To znamená i aplikace, kterou se zabývá tato práce.

2.2 Aplikace

Aplikace pro mobilní zařízení se sestává z několika základních stavebních prvků. Mezi tyto prvky patří Aktivity, Služby, Content Provider, Broadcast Reciever a Intents. Podrobněji jsou tyto prvky vysvětleny níže. Informace v této kapitole jsou čerpány z publikace Allena Granta (2013) [14]

Aktivita

Pod pojmem Aktivita si můžeme představit vizuální reprezentaci aplikace, jejíž rozložení lze definovat pomocí XML layoutu (definuje rozložení prvků) nebo je vytvořena dynamicky při vytváření aplikace[5]. V jedné aplikaci se pak může nacházet více aktivit, přičemž aktivní může být vždy pouze jedna. Z jedné aktivity je možné vyvolat jinou, přičemž původní aktivita je pozastavena a vložena na zásobník aktivit a je ji možno vyvolat pomocí tlačítka zpět. Aktivity se mohou nacházet v několika stavech (aktivní, pozastavená, zastavená, zničená), které jsou souhrnně označovány jako *životní cyklus aktivit*[2].

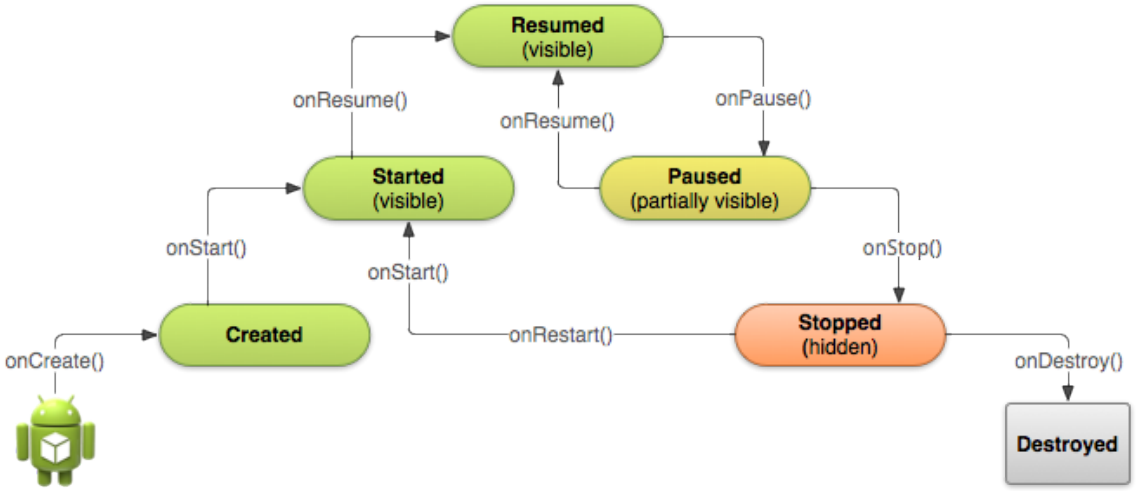
- **onCreate()** – po prvním spuštění nebo obnovení ze zásobníku aktivit je volána metoda *onCreate()*, ve které probíhá vytvoření layoutu, nastavení proměnných a prvků UI. V případě, že aktivita již byla aktivní, může být její stav uložen v objektu *Bundle*, který je předávám jako parametr této metody.

- **onStart()** – po metodě *onCreate()* následuje metoda *onStart()*[6]. Ta je zavolána ve chvíli, kdy se stává aktivita pro uživatele viditelnou. Další v pořadí je *onResume()*,

[4] Dále může být tato přeložená část uchována v cache a znovu použita, avšak toto uložení se vykoná až po mnohonásobném překladu stejné části kódu. Zbylé části kódu jsou nadále překládány opakovaně.

[5] Obě tyto metody lze kombinovat.

[6] Po metodě *onStart()* může také následovat po metodě *onRestart()*.



Obrázek 2.2: Životní cyklus aktivity.[6]

která je volána, jakmile je aktivita v popředí. Zde je aktivita schopna přijímat uživatelské vstupy a je vhodné zde otevírat výlučná spojení (například s databází, soubory, a podobně) a nebo začínat animace, které spouští pro lepší interakci s uživatelem. Po této metodě musí vždy následovat metoda *onPause()*, která je volána, jakmile se aktivita přesune do pozadí.

- **onPause** – tato metoda musí uvolnit všechny výlučná spojení, ukončit všechny animace, a také by se měla postarat o všechna neuložená data. Souhrnně se dá říci, že by tato metoda měla ukončit vše, co započala předchozí metoda *onResume()* a měla by se postarat o uložení všech vytvořených dat.
- **onRestart(), onStop() a onDestroy()** – při pohledu do schématu 2.2 je vidět, že už zbývají vysvětlit pouze poslední tři metody. Metoda *onRestart()*, která je volána před znovuspuštěním aktivity, metoda *onStop()*, volána pokud není aktivita viditelná a metoda *onDestroy()*, jakožto poslední metoda před ukončením aktivity. Důležité je, že tyto poslední 3 metody nemusí být nikdy zavolány. Proto by se například metoda *onStop()* neměla využívat k ukládání neuložených dat, protože by mohlo dojít k jejich ztrátě.

Služby

Služby jsou komponenty, které provádí dlouho běžící akce. Tyto komponenty jsou určeny především k běhu na pozadí bez uživatelského rozhraní a ovládány jsou z jiných komponent, většinou aktivit. Službu může využívat jedna nebo i více aktivit, čímž může docházet k meziprocesové komunikaci (*IPC – Inter process communication*).

Widged

Jedná se o zmenšeninu aplikace, která je určena pro vložení do jiné aplikace, nejčastěji na domovskou stránku. Widgedy jsou periodicky obnovovány a jejich hlavním účelem je rychlý přístup k hlavním funkcím aplikace, která není přímo otevřena. Využívány jsou

u multimediálních přehrávačů, kalendářů, emailů a u mnoha dalších aplikací.[7]

Content Provider

Content Provider poskytuje standardní způsob, jakým lze přistoupit k datům z jiného procesu. Tato komponenta řídí přístup k datům tak, aby se programátor nemusel zajímat, jak jsou fyzicky uložena a zároveň zajišťuje jejich zabezpečení.

Intent

Komunikace mezi jednotlivými komponentami je v Androidu implementována pomocí zasílání zpráv, tj. objektů označovaných jako *Intent*[8]. Základním využitím tohoto mechanismu je například spuštění nové aktivity nebo služby. Stejně tak jsou ale využívány k odeslání výsledku nějaké operace a nebo informování o aktuálním stavu (příchod SMS, připojení USB kabelu, vložení SD karty a mnoho dalších). Veškerá komunikace probíhá prostřednictvím těchto objektů a proto se jimi dá informovat opravdu o čemkoli. *Intenty* se dělí na dva typy:

- *Implicitní intenty* – jsou určeny pro konkrétní komponentu a označení této komponenty je jejich součástí.
- *Explicitní intenty* – specifikují pouze akci, která se má vykonat, nikoli komponentu, která ji má vykonat. Jedná se například o žádost otevření webové stránky, přičemž volba prohlížeče je pouze na uživateli.

Pro tuto potřebu se zavedly tzv. *Intent filtry*, uvedené v *manifestu* aplikace[9], které specifikují jaké typy implicitních intentů jsou přijímány kterými komponentami. Intent filter je možné zadat například pomocí *akce* (udává akci která se má následně vykonat) a *kategorie* (udává kategorii intentu), přičemž povinná je pouze položka *akce*.

Často je možné se setkat s intent filtrem, který umožňuje spuštění hlavní aktivity aplikace pomocí ikony. Tento filtr má definovány právě položky *kategorie* a *akce* a vypadá následovně:

```
<activity android:name="SampleActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

View

Komponenta[10] je základní prvek při vytváření uživatelského rozhraní pro systém Android. Každá komponenta je třída zděděná od základní třídy View nebo ViewGrop. V systému je pro vývojáře definováno několik komponent (Button, TextView, ListView, RadioButton,

[7] V této práci bude označení *widged* využíváno také pro vložení graficky zpracované dodatečné informace do videa.

[8] Do češtiny se toto označení většinou nepřekládá, nicméně v některé literatuře je možné narazit na termín *úmysl*.

[9] Manifest aplikace je XML soubor definující základní vlastnosti a nastavení aplikace.

[10] V této kapitole je komponenta myšlena jako prvek grafického uživatelského rozhraní.

Linear Layout ...), které stačí jen použít, případně upravit dle potřeby. Stejně tak jako jsou definovány tyto komponenty, je možnost vytvoření své vlastní, ať už zděděné od samotné třídy View nebo od jiné třídy již zděděné.

Třída View má několik základních metod[3], které je možné podle potřeby redefinovat. Zde jsou uvedeny některé z nich.

- **onDraw()** – této metodě je předán Canvas, na který je možné vykreslovat to, jak má komponenta vypadat. Může se jednat o jakoukoli 2D grafiku jako je text, geometrické objekty, čáry apod. Tato metoda se vyvolá implicitně při vykreslování celého uživatelského rozhraní a nebo explicitně, po zavolání metody **invalidate()**.
- **onMeasure()** – jako parametry této metody jsou předány šířka a výška, který byly této komponentě přiděleny rodičem podle vykreslení ostatních prvků na obrazovce.
- **onTouchEvent()** – metoda, která je volána při dotyku na tuto komponentu, tj. na místo na obrazovce, které zabírá. Zde je možné definovat akce, které se mají vykonat v závislosti na dotyku. Může se například jednat o uložení do databáze, vyvolání aktivity, otevření souboru a podobně.

Kompletní seznam metod můžeme nalézt na oficiálních stránkách Android Developers[3].

Pro vytvoření vlastní komponenty je nutné vytvořit novou třídu, která rozšiřuje třídu View. Dále se redefinují metody, podle požadavků na novou vlastní třídu. Vložení této nové třídy může proběhnout buď dynamicky, nebo vložením XML kódu do layoutu.

Pro příklad je zde uvedeno vytvoření vlastní třídy se jménem `CustomView`, která je umístěná v balíku `com.example.app`, se dvěma vlastními atributy[11] (`stripedBackground` a `numbersOfVisibleChild`) vypadá XML kód pro vložení následovně:

```
<com.example.app.CustomView
     custom:stripedBackground="true"
     custom:numbersOfVisibleChild="6" />
```

2.3 Senzory

Tato kapitola se zabývá průzkumem senzorů dostupných na mobilních zařízeních k pozdějšímu návrhu reálných widgedů. Mobilní zařízení mají k dispozici velké množství senzorů pro různé použití. Tyto senzory lze rozdělit do 3 základních částí:

- pohybové senzory,
- senzory okolního prostředí,
- senzory pozice.

Vzhledem k cíli této práce zde budou rozvedeny pouze senzory pohybové a senzory prostředí.

[11] Vlastní atributy zadané v XML se musí nejprve definovat v souboru *res/values/attrs.xml*

Pohybové senzory

Android nabízí několik pohybových senzorů, které se uživateli mohou zdát jako reálné hardwarové komponenty, nicméně nemusí tomu tak být. U většiny dnešních mobilních zařízení jsou běžně k dispozici dva hardwarové senzory. Jedná se o akcelerometr a gyroskop. Akcelerometr má za úkol sledovat zrychlení. Zde je potřeba zdůraznit, že se jedná skutečné jen o zrychlení, nikoli o rychlost. Tento senzor dokáže tedy zmonitorovat pouze změny rychlosti mobilních zařízení. Druhý zmíněný senzor, gyroskop, slouží k určení natočení mobilního zařízení, přičemž sleduje osy x, y, z.

Další senzory, které Android nabízí, jsou většinou softwarově dopočítávány z těchto dvou zařízení. Nicméně u nejnovějších zařízení se začínají některé další senzory vyskytovat také hardwarové. Dále je tedy možné použít senzory gravitace, lineárního zrychlení nebo vektorů rotace.

Senzory pozice

Stejně jako tomu bylo u pohybových senzorů, tak i zde platí, že všechny senzory nejsou hardwarové. Hardwarovými jsou v této skupince pouze senzory geomagnetického pole a proximity senzor. Pomocí prvního senzoru je možné detekovat magnetické pole, což znamená, že jej lze využít například pro tvorbu kompasu. Proximity senzor má oproti tomu pouze jednostranné využití a to detekci blízkosti předmětu. Tento senzor dokáže vracet pouze booleovskou hodnotu a jako jediný ze senzorů je tzv. *interrupt-based*[12], nikoli *pull-based*[13] jako ostatní.

2.4 ORM Lite

V OS Android je k dispozici relační databázový systém SQLite [16]. SQLite je pouze malá knihovna psaná v jazyce C, přičemž každá databáze je uložena v samostatném souboru s koncovkou *.dmb*. Tato databáze je veřejně šiřitelná (public domain) a podporována různými programovacími jazyky (Java, C/C++, Perl, Python, ...) i operačními systémy (Linux, Mac OS a Windows).

ORMLite (Lightweight Object Relational Mapping) umožňuje ukládání Java objektů do SQL databáze. Java objekty je nutné označit pomocí Java anotace. Zde je uveden příklad takového označení. Jedná se o jednoduchý objekt `Osoba`, obsahující dva atributy `jmeno` a `prijmeni` a položku `_id`, jakožto identifikátor objektu a tedy i záznamu v relaci. Tento objekt je namapován na stejnojmennou relaci se stejnými atributy jako Java objekt.

```
@DatabaseTable(tableName = "Osoba")
public class Osoba {
    @DatabaseField(id = true)
    private int _id;

    @DatabaseField
    private String jmeno;

    @DatabaseField
```

[12] Senzor typu interupt-based informuje o svém stavu přerušením, tj. procesor nemusí cyklicky načítat hodnotu z tohoto senzoru.

[13] Procesor cyklicky načítá hodnotu z těchto senzorů.

```
    private String prijmeni;

    Osoba() {
     //prazdny konstruktor
    }
}
```

Samotné vytvoření tabulky se pak provede pomocí tohoto příkazu v metodě *onCreate()* třídy zděděné od *OrmLiteSqliteOpenHelper*:

```
TableUtils.createTable(connectionSource, Person.class);
```

2.5 Nástroje pro vývoj

Při výběru vhodných nástrojů pro tento operační systém se nabízí hned několik možností. Všechny jsou veřejně dostupné a je na každém, co mu bude vyhovovat. Nejjednodušší možností, která je k dispozici je možnost využít oficiální IDE Android Studio, které je postavené na základech IntelliJ IDEA a obsahuje všechny potřebné komponenty. Druhá možnost je využití balíčku SDK a jeho připojení například k vývojovému prostředí Eclipse.

Android SDK

SDK obsahuje všechny potřebné nástroje pro vývoj aplikací. Důležitou součástí je ADT (Android Developer Tools), které rozšiřuje vývojové prostředí Eclipse pro vývoj mobilních aplikací. Další součástí balíčku SDK je emulátor virtuálních zařízení (včetně specifikace reálných zařízení), díky kterému je možné vyvíjet aplikace i na přístrojích, které nejsou fyzicky k dispozici. Těmto zařízením je také možné měnit hardwarovou specifikaci a nebo simulovat různé situace, které mohou nastat (slabý signál, stav baterie, apod.) pomocí Android Virtual Divice Manager. Emulátor je také schopen simulovat běh dvou zařízení současně, které spolu mohou komunikovat.

Kapitola 3

Knihovny pro práci s multimédii

3.1 FFmpeg

FFmpeg [8] je framework umožňující práci s multimediálním obsahem jako je například video nebo audio. Tento framework nabízí velkou škálu funkcionalit, které umožňují úpravu multimediálních souborů od kódování a dekódování, komprese videa, až po nebo práci s filtry. Software je vyvíjen skupinou *FFmpeg team*, která jej dává k dispozici jak v podobě zdrojových kódů, tak přeložených souborů a to pod licencí Lesse General Public License (LGPL) verze 2.1. Ovšem některé části jsou licencovány jako GNU General Public License (GPL) version 2.

Mezi velké výhody této knihovny lze zařadit multiplatformní použití. FFmpeg lze proto použít na řadě architektur, ať už na běžných počítačích, nebo na zařízeních s omezenými zdroji (mobily, tablety a další). Samozřejmostí je podpora architektur ARM, x86, MIPS a další. Stejně tak je tomu s podporovanými operačními systémy, kde je možné nalézt Linux, FreeBSD, OpenBSD, Microsoft Windows a další.

Jako další velká výhoda je podpora celé řady kodeků a kontejnerů multimediálního obsahu. Podporovaných formátů je celá řada, avšak u některých z nich se jedná o podporu pouze pro čtení nikoli pro zápis. Pro přehled je zde uvedeno několik podporovaných formátů pro video, audio a obrázky:

- **video:** MPEG-2, MPEG-4, H.264, Windows Media Video (7, 8), RealVideo (1.0, 2.0),
- **audio:** AAC, FLAC, MPEG, Windows Media Audio (1,2),
- **obrázky:** JPEG, PNG, BMP, TIFF.

Kompletní seznam je uveden na oficiálních stránkách FFmpegu[8].

Pro zajímavost je zde uvedeno několik aplikací, které využívají FFmpeg. Mezi nejznámější lze zařadit Blender, VLC, MPlayer nebo Google Chome.

3.2 Knihovny pro práci s multimédii pro Android

V kapitole výše bylo uvedeno, že knihovna FFmpeg je k dispozici pro použití i na systémech Android, a proto by se dalo říci, že již byla probrána. Ovšem tato kapitola je spíše koncipována jako přehled knihoven, které lze používat přímo v jazyce Java. Jedná se o knihovny, které zapouzdřují samotnou knihovnu FFmpeg, a nebo, v případě *android.media*, jsou nabízeny jako standardní funkce systému.

JavaCV

JavaCV[10] využívající knihovnu JavaCPP, umožňuje volání nativních funkcí z knihoven pro počítačové vidění a abstrahuje je do větších modulů pro snadnější použití. Mezi zastřešené knihovny patří OpenCV, libdc1394, PGR FlyCapture, OpenKinect, videoInput, ARToolKitPlus, flandmark, ale také pro tento projekt důležitá knihovnu FFmpeg. JavaCV nabízí možnost jednoduššího použití funkcí těchto knihoven přímo z projektů psaných v programovacím jazyce Java, přičemž uživatel má možnost si importovat pouze ty části tohoto projektu, které potřebuje použít. Knihovnu JavaCV proto lze chápat jako mezivrstvu mezi vytvářenou aplikací a knihovnami zmíněnými výše.

Tato knihovna je stejně jako FFmpeg šířena pod licencí GNU General Public License verze 2 a je vedena a spravována Samuelem Audetem. Přesto, že je knihovna vyvíjena především tímto člověkem, je knihovna velmi rozsáhlá a využívá ji velké množství vývojářů. Nevýhoda je ovšem v absenci dokumentace a tak jediný způsob jak zjistit způsob použití této knihovny je z několik příkladů, z čehož 2 jsou přímo pro mobilní systém Android, a nebo ze samotných zdrojových kódů této knihovny, které jsou volně k dispozici. Nutno dodat, že existuje internetové fórum vývojářů využívajících tuto knihovnu, kde je možnost najít mnoho odpovědí, přesto tato možnost nedokáže vyvážit absenci dokumentace.

Jjmpeg

Jjmpeg[11] je projekt podobný knihovně JavaCV s tím rozdílem, že jjmpeg zastřešuje pouze knihovnu FFmpeg. Opět se jedná o projekt, který lze použít na platformě Android a vytváří určitou nadstavbu nad knihovnou FFmpeg pro snadnější použití. Konkrétně pro mobilní systém Android je k dispozici upravený projekt *jjmpeg-android*. Bohužel jak projekt jjmpeg, tak jjmpeg-android se od roku 2013 dále nevyvíjí. Jjmpeg je šiřitelný pod licencí GNU General Public License verze 3.

Android.media

Prostřednictvím balíku android.media[5] dříve nabízel Android pouze možnosti pro přehrávání, ale nikoli pro editaci jednoho nebo více video souborů najednou. Tato situace se změnila s příchodem API verze 21 (Android 5.0), kde byly přidány nové a upraveny některé stávající možnosti tohoto balíku.[1] V současnosti již android.media nabízí možnosti jak získávat či vkládat snímky do video souboru, nebo jak pracovat podobným způsobem s audio souborem. Přesto je pro složitější práci s více soubory a pokročilejšími funkcemi vhodnější využít knihovnu FFmpeg přes některé její rozhraní z jazyka Java.

Diplomová práce Ing. Tomáše Slavotínka

Tato práce[18] vznikla na Fakultě informačních technologií Vysokého učení technického v Brně jako diplomová práce. Autor vytvořil knihovnu, která zastřešuje knihovnu FFmpeg a umožňuje její používání z jazyka Java prostřednictvím tříd *MediaFile* a *MediaLib*. Tyto dvě třídy obsahují Java Native Interface (JNI), pomocí kterého je navázána část knihovny psaná v jazyce C a C++. V nativní části knihovny pracuje autor přímo s knihovnou FFmpeg.

[1] Kompletní seznam změn je zveřejněn zde: https://developer.android.com/sdk/api_diff/21/changes.html

Při použití této knihovny dochází k otevření souboru a vytvoření instance třídy *MediaFile*. V tomto objektu je provedena potřebná inicializace, jsou otevřeny jednotlivé streamy tohoto souboru a zjištěny dodatečné informace o souboru jako je například jeho délka. Pomocí této instance je možné soubor přehrávat, posouvat se v čase, nebo získávat a ukládat snímky videa. Tyto požadavky jsou posílány jádru knihovny, psaném v jazyce C a C++, a později je navrácen výsledek operace. Při hledání samotného principu fungování této knihovny je tedy nutné zkoumat přímo kód v jazyce C a C++, nikoli třídu *MediaFile*.

K této knihovně opět schází dokumentace, ale alespoň některé její části jsou vysvětleny v technické zprávě této diplomové práce.

Kapitola 4

Přehled existujících řešení

Aplikací zabývajících se zpracováním videa je celá řada. Některé z nich nabízí pouze omezené úpravy, ale existují také komplexnější aplikace, které umožňují více možností editace videa, zvuku nebo přidávání widgedů, jakožto textu, fotek, obrázků a jiných. V následujícím textu bude popsáno několik různých aplikací, které tyto rozdíly ukazují. Aplikace byly získány v oficiálním obchodě Google Play[9].

4.1 VideoEditor

Jednoduchý VideoEditor umožňující základní úpravy videa. Umožňuje přidávat hudbu nebo fotografie do videa, dále je možno video zkrátit a nebo naopak sloučit více videí do jednoho za sebe. Mezi jednotlivá videa nelze při slučování vkládat žádné přechody. Dále také není možné videa nijak upravovat (Nezle aplikovat filtry pro úpravu obrazu, nebo měnit velikost videa.). Podobně není možné upravovat audio stopu kromě jediného nastavení a to hlasitosti přidaného mp3 audia souboru.

Z pohledu grafického uživatelského rozhraní jde o jednoduchou přehlednou aplikací, kde se v první fázi vybere cílená funkcionalita, poté zdrojové soubory a nakonec je možno video uložit.

Výsledné video se dá uložit pouze v jediném formátu mp4, ale se dvěma předdefinovanými rozlišeními (*default resolution* – ponechá video v jeho rozlišení, *instagram resolution* – rozlišení 480 $\times$ 480 px). Zajímavá funkcionalita je vložení videa na některou ze sociálních sítí přímo z aplikace.

4.2 VideoShow

VideoShow nabízí mnohem více možností editace, než předchozí aplikace. Kromě základních funkcí, které mají obě aplikace stejné, je zde možnost vkládat přechody mezi videa nebo fotky, vkládání audio záznamů nejen od uživatele, ale i univerzálních melodií z knihovny aplikace, dále vkládání textu a ikonek, u kterých je modifikovatelná velikost a rotace.

Aplikace nabízí o několik funkcí více, než předchozí, ale přesto se tvůrcům podařilo dosáhnout přehlednějšího uživatelského rozhraní. Po importování médií, které se budou zpracovávat, a po vložení libovolného elementu (text, obrázek, foto, ...) se zobrazí časová osa pouze s tímto jedním vybraným elementem a tudíž odpadá nepřehlednost při práci s více vkládanými elementy. Nevýhoda ovšem je, že pro editaci již vložených elementů,

musíte upravovat každý zvlášť bez možnosti porovnání s jiným. Tento problém je nepříjemný zejména ve chvíli, kdy je potřeba vložit dva elementy v jeden časový okamžik.

Export výsledného videa je opět možný pouze ve formátu mp4 a ve dvou rozlišeních: *HD mode* – 800 × 448 px a *Fast mode* – 736 × 416 px. Výsledné video je možné stejně jako v předchozí aplikaci vložit na některou ze sociálních sítí.

4.3 Cute CUT

Aplikace zastupující nejkomplexnější video editory pro mobilní zařízení. Kromě základních funkcí zpracování video a audio souborů je zde možnost změny barvy pozadí, možnost práce s vrstvami, kreslení na plátno a to různými štětci a tvary. Dále je možnost měnit průhlednost, barvu, šířku čáry a u obrázků a videí možnost změny rotace, velikosti, viditelnosti a přidání stínu.

Přes svou komplexnost ovšem strádá na rychlosti[1] a také uživatelské rozhraní není tolik přehledné.

Nevýhodou je její použitelnost pouze na tabletech.

4.4 Magisto

Aplikace Magisto se nejvíce podobá cíli, ke kterému je směřována i tato vyvíjená aplikace. Kromě editace audio a video souborů, teoreticky v neomezených množstvích, nabízí možnost automatického vytvoření krátkých videosekvencí z jednoho velkého celku. Dále je možnost tyto krátké sekvence z různých videí automaticky zkombinovat a nechat si automaticky vytvořit jeden výstupní soubor. Aplikace samozřejmě nezajistí, aby ve videu byly ty nejlepší momenty, ale pokud jsou dány aplikaci k dispozici vhodné nahrávky, pak je výsledek velmi solidní.

Samozřejmostí je vkládání hudby a fotek a navíc, je možnost používat styly výsledných souborů. Těmito styly jsou myšleny barevné kombinace, které jsou aplikovány jako barevné filtry na výsledné soubory. Aplikace navíc zvládá stabilizaci videozáznamů a rozpoznávání obličejů.

[1] Aplikace byla zkoumána na zařízení iPad 3. generace, kde docházelo k nepříliš plynulému zobrazování a reagování na uživatelské interakce.

Kapitola 5

Návrh

Vytvoření aplikace má několik zásadních částí, které musely být řešeny. Nejprve musela být vyřešena funkčnost celé aplikace, tedy především vyřešení otázek:

- Co bude aplikace umět?
- Jak se s ní bude pracovat?
- Co bude možné editovat a vkládat?
- A další.

Podle specifikovaných požadavků na funkčnost byly vybrány potřebné technologie a dále bylo navrženo, jak by mohla aplikace vypadat a jak budou jednotlivé funkčnosti nabídnuty uživateli tak, aby aplikace nebyla pro uživatele složitá. Tímto se postupně vytvořil kompromis mezi tím, co aplikace může všechno umět a kolik je toho ještě možné uživateli přijatelně zobrazit. Následně se mohlo přistoupit k navržení schéma databáze, do které se budou potřebná data ukládat a začít s implementací.

5.1 Cíl

Cílem bylo vytvoření mobilní aplikace umožňující editaci video a audio souborů. Dále bylo požadováno zasadit do videa widgedy, které reprezentují nasbírané data s GPS, akcelerometru, gyroskopu a dalších senzorů mobilních zařízení. Aplikace by měla pracovat i s více mediálními soubory najednou a výsledný projekt exportovat. Aplikace by tedy měla umožňovat následující úkoly:

- Pracovat se soubory typu video, audio a senzor.
- Vytváření interních projektů aplikace.
- Vložení video souboru do aplikace.
- Připojení senzoru (widgedu) k video souboru. Tyto widgedy by měly data ze senzorů reprezentovat v uživatelsky přívětivé podobě.
- Připojení audio souboru k videu.
- Zkrácení/rozdělení videa, audia nebo souboru s daty ze senzoru.

- Připojení více souborů stejného typu za sebe a vytvoření sekvence těchto souborů.
- Uložení dokončených i nedokončených projektů do databáze, aby byla možná jejich další editace.
- Export projektu v různých rozlišeních videa.
- Nabídnou uživateli možnost vizualizace snímku výsledného videa ve zvoleném čase.
- Přehrát projekt bez jeho předchozího exportu.
- Nastavení stylu projektu.
- Styl ovlivní reprezentaci widgedu.

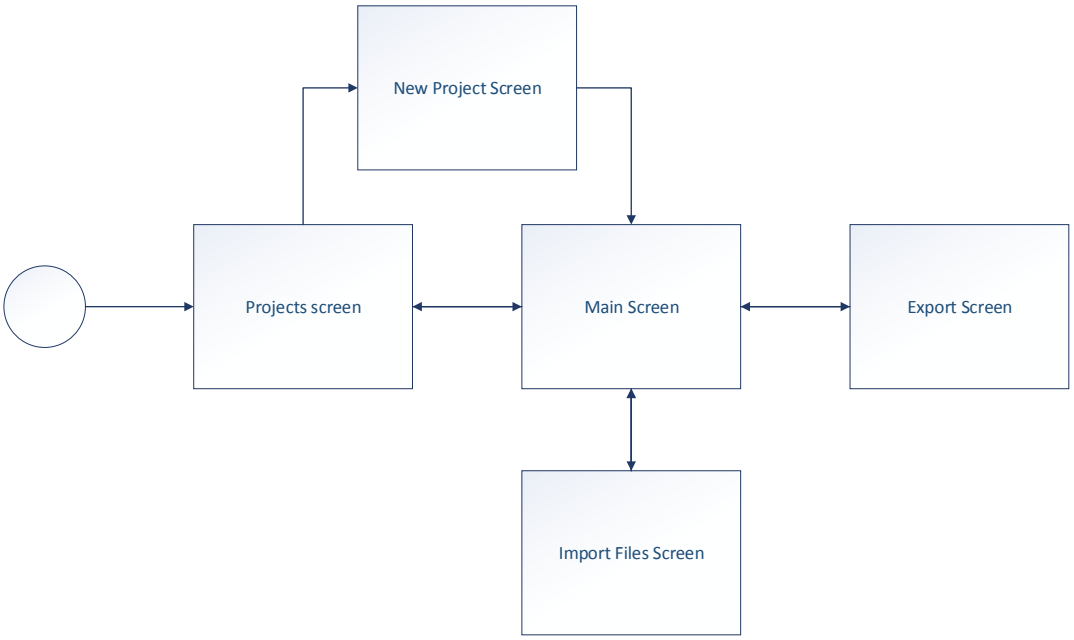
5.2 Výběr technologií

Podle specifikace požadavků bylo důležité vybrat vhodné nástroje pro realizaci aplikace. Nejdůležitější částí bylo vybrání vhodné knihovny na zpracování video a audio souborů. Vzhledem k absenci dokumentace u většiny prezentovaných knihoven v kapitole 3.2 nebyl výběr nijak jednoduchý. Nakonec byla vybrána knihovna Ing. Tomáše Slavotínka i proto, že vznikla jako diplomová práce na Fakultě informačních technologií Vysokého učení technického v Brně a byla k ní k dispozici technická zpráva, která dokumentovala alespoň některé její části. Později se však ukázalo, že knihovna nebyla nejvhodněji zvolena, protože byla vytvářena především pro překódování videí do jiných formátů a jejich přehrávání. V případě některých specifických požadavků této vytvářené aplikace, které jsou krátce popsány v kapitole 6.5, knihovna nefungovala vždy správně a některé požadované úkoly nedokázala vůbec vykonat.

Dále byla zvolena minimální verze OS Android, tedy API, pro které je aplikace podporována. Z tabulky 2.1, je viditelné že 93,2 % uživatelů využívá API verze 15 nebo novější a proto bylo zvoleno toto API.

5.3 Návrh aplikace

Po specifikaci požadavků bylo možné přejít k návrhu struktury aktivit mobilní aplikace. Jedna aktivita představuje zjednodušeně řečeno obrazovku, na které jsou umístěny ovládací prvky. Struktura obrazovek je naznačena v obrázku 5.1. Po spuštění aplikace bude spuštěna aktivita *Projects Screen* a z ní bude možné založit nový, nebo otevřít uložený projekt. Při vytvoření nového projektu bude spuštěna aktivita *New Project Screen*, ve které budou nastaveny potřebné údaje nového projektu. V opačném případě bude spuštěna hlavní aktivita aplikace (*Main Screen*), která bude následovat i po aktivitě *New Project Screen*. Z této hlavní aktivity se nelze vrátit zpět do aktivity *New Project Screen*, nýbrž pouze do aktivity *Projects Screen*. Je to z důvodu uložení nového projektu do databáze přímo po jeho vytvoření a tudíž je možno jej pouze upravovat, nikoli se vrátit zpět a znova jej vytvořit. Z hlavní aktivity může být spouštěna aktivita *Imported Files Screen*, sloužící pro import souborů do aplikace. Dále je možné spustit obrazovku *Export Screen*, přičemž zde se nejedná o aktivitu, nýbrž pouze o dialog. Pro ilustraci je v diagramu tento dialog zobrazen stejně jako aktivity.



Obrázek 5.1: Obrazovkový tok aplikace.

Widgedy, styly, data ze senzorů

V aplikaci byly navrhnuty styly, které ovlivňují zobrazení widgedů. Při dalším vývoji tyto styly mohou ovlivňovat i jiné média nebo nastavení projektu. V současném návrhu má uživatel možnost zvolit si jeden ze stylů *Sky Diving* nebo *Snow*, který kromě vizuálního vzhledu widgedu definuje zobrazení typu dat ze senzorů. Styl *Sky Diving* je určen pro reprezentaci widgedu nadmořské výšky, oproti tomu styl *Snow* je určen pro reprezentaci widgedu rychlosti.

Data, která jsou vytvořena v této práci pro demonstraci aplikace, nejsou skutečná, nýbrž uměle vygenerovaná[1], protože například data ze seskoku s padákem jsem neměl možnost nasbírat. Nicméně vytvořená data jsou podepřena reálnými hodnotami. Například pro styl *Sky diving* byla data vytvořena podle výšky, ze které se skáče z letadla, pro rychlost jakou se padá apod.

Samotné widgedy je možné umístit do 4 pozic ve videu. Jedná se o umístění do libovolného rohu videa. Widgedy jsou navrhnuty jako přidané informace do videa a proto není vhodné je umísťovat například doprostřed, kde by na sebe vázaly příliš velkou pozornost.

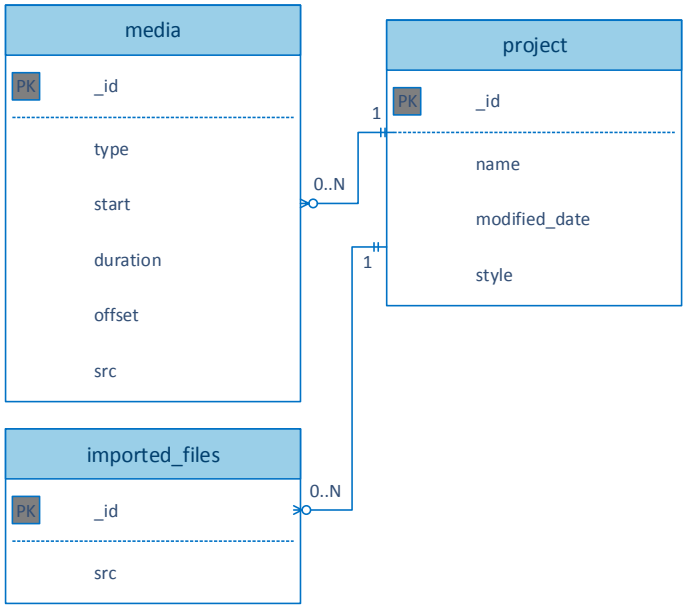
5.4 Reprezentace dat

Pro uchování dat v perzistentním stavu byla navržena jednoduchá databáze, ve které jsou uchovávány vytvořené projekty. U každého projektu je ukládáno jméno, datum poslední změny a styl. Dále jsou uloženy všechny importované soubory (v tabulce *imported_files*) a média, která byla použita na časové ose projektu[2]. U jednotlivých médií je nutné si pamatovat umístění na časové ose, jejich délku, offset a zdroj. U importovaných souborů je

[1] Script použitý pro vytvoření těchto dat je součástí přiloženého DVD.

[2] Z importovaného souboru se vytvoří média sekvence na časové ose právě přesunem na časovou osu

uchován pouze zdroj. U každého projektu pak může být nula nebo více médií i importovaných souborů. Význam importovaných souborů a médií je vysvětlen v následující kapitole 5.4.



Obrázek 5.2: ER diagram databáze

Data ze senzorů jsou uchovávána v textových souborech ve formátu JSON jako kolekce dvojic. Jako první hodnota ve struktuře je uvedeno časové razítko a jako druhá je hodnota senzoru. Po zpracování jsou tyto soubory převedeny do struktury, kde jsou jednotlivé hodnoty seřazeny právě podle časového razítka. Příklad takovéhoto souboru je uveden v následujícím příkladu:

```
{"0":"20","1":"30","2":"40","3":"45","4":"50","5":"55"}
```

Tento příklad ukazuje nasbíraná data v časovém intervalu 0-5 s, kde krok vzorkování hodnoty senzoru je 1 s. Nasbírané hodnoty v jednotlivých časech jsou v rozmezí hodnot 20-55.

Práce se soubory a médii

Při vytváření výsledného projektu je nutné nejprve importovat zdrojové soubory a uchovat si je k dispozici pro další využití. K tomuto účelu byl navržen způsob uchování dvou typů objektů a to *Imported Files* a *Media*. Po výběru souborů ze složky zdrojových souborů se uchová tento importovaný soubor. Při pozdějším využití tohoto souboru, tzn. při použití na časové ose projektu, dojde k vytvoření objektu typu Media, který si uchová stejný zdroj (položku *src*) jako objekt, ze kterého byl vytvořen. Objekt typu média následně otevře tento soubor pro získání dat. U souborů typu Video a Audio dojde k vytvoření objektu typu *MediaFile*[3] a podle něj doplní položku *duration*, která má význam délky tohoto souboru.

Při ukládání těchto médií se uchovává pouze zdroj k video souboru a je tedy nutné znovu otevřít soubor, tedy vytvořit instanci třídy *MediaFile*.

[3]Význam tohoto objektu je popsán v kapitole 6.5

5.5 Uživatelské rozhraní

Po nalezení kompromisu ve funkčnosti aplikace a její jednoduchostí, bylo navrženo uživatelské rozhraní, které bylo později mírně upraveno, nicméně podstatné části zůstaly zachovány. Zde budou popsány všechny části z diagramu 5.1 včetně myšlenek, které k danému grafickému řešení vedly. Jednotlivé aktivity z tohoto diagramu budou dále popisovány jako obrazovky, které budou nabízeny uživateli.

Projcts Screen

První obrazovka zobrazující se ihned po startu aplikace je *ProjectsScreen*. Na této aplikaci je uživateli nabídnuta možnost vytvoření nového projektu, prostřednictvím tlačítka *Create new project*, a možnost otevřít jeden z dříve vytvořených projektů. Tyto projekty jsou reprezentovány v seznamu, přičemž u každého projektu je zobrazen jeho název a datum poslední změny.

New Project Screen

Tato obrazovka má za úkol získat informace o nově vytvořeném projektu. Konkrétně se jedná o název a styl projektu. Datum poslední modifikace zmíněné v předchozím textu není uživatelem zadáváno, ale přidáváno automaticky aplikací. Nachází se zde také tlačítko pro potvrzení údajů a vytvoření projektu.

Main screen

Tato obrazovka je nejdůležitější z celé aplikace. Zde bylo potřeba zajistit intuitivní ovládání pro editaci videa a proto byl navržen základní koncept této obrazovky, který se při běhu aplikace nijak nemění. Základem této obrazovky je komponenta dále nazývaná jako *časová osa*, kterou představuje pruh obrazovky při její spodní části, na který jsou umísťovány jednotlivé média (viz. 5.4). Tato komponenta je shora označena *časovým měřítkem*, nad kterým se nachází tzv. *posuvník* (ukazatel aktuální pozice v čase). Celá tato komponenta se dá zvětšovat/zmenšovat (dochází ke zvětšení/zmenšení zobrazovaného časového úseku projektu) a posouvat v čase do stran. Časová osa obsahuje 3 vrstvy. Každá z vrstev je určena jen pro jeden typ médií a médium nejde vložit jinam, než do vrstvy pro něj určené. První shora je vrstva video souborů, druhá je vrstva audio souborů a třetí je vrstva senzorů. Vrstvy jsou označeny červenou, modrou a zelenou barvou.

Nad časovou osou je umístěn náhledový box aktuálního snímku. Pomocí posuvníku se lze posouvat v čase projektu a aktuální snímek z daného času je vykreslen právě v tomto náhledu. Aktuální snímek je vykreslován již takový, jaký by byl v případném vyexportovaném souboru, tedy je zde složeno video s případným widgedem. Vedle tohoto náhledu je umístěno tlačítko pro spuštění přehrávání nebo naopak pauzy. Přehrávání probíhá opět v rámci tohoto náhledového boxu.

Dalším požadavkem byla přístupnost importovaných souborů pro jejich použití v projektu a také možnost nastavení jak samotného projektu tak i jednotlivých typů médií. Toto bylo vyřešeno dvěma postranními menu z obou stran obrazovky. Obě menu budou defaultně schovány a uživatel je může vysunout v případě potřeby. Levé menu (dále označováno jako *menu souborů*) obsahuje seznam importovaných souborů a pravé menu (dále označováno jako *menu nastavení*) nastavení k právě vybranému médiu.

Levé menu tedy obsahuje seznam importovaných médií a tlačítko pro přechod do aktivity k importu dalších souborů. Obsah tohoto menu se nebude nijak měnit, vyjma přímého importu dalších souborů.

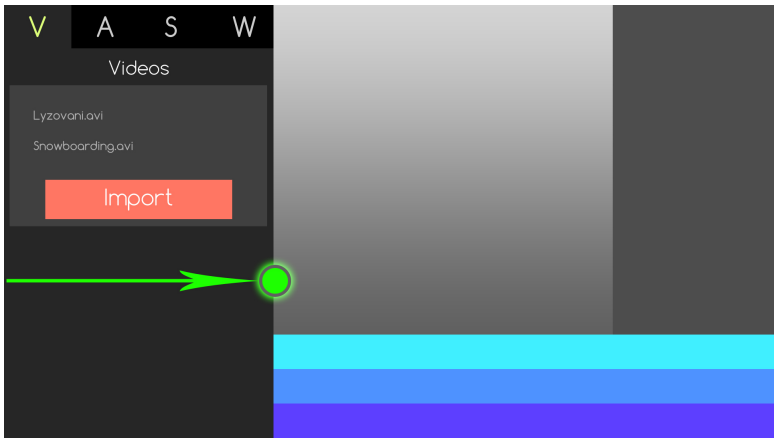
Oproti tomu pravé menu bude obsahovat nastavení právě vybraného média, nebo globální nastavení projektu, které bude zvoleno defaultně. Změna typu nastavení zobrazená v tomto menu bude provedena po krátkém stisku média na časové ose. Pokud bude kliknuto kdekoli mimo některé z médií, bude opět zobrazeno globální nastavení.

Poslední důležitou částí této obrazovky je **ActionBar** [15]. Actionbar je systémové menu aplikace, které umožuje dynamické vkládání položek. V tomto projektu je Actionbar využit jako paleta s nástroji na úpravu. Stejně jako se mění obsah *menu nastavení* se mění i obsah Actionbaru. Tedy například při výběru média typu video se v Actionbaru objeví nástroje pro úpravu videa.

Ovládaní

Ovládání této obrazovky probíhá pomocí gest. Zde jsou uvedeny příklady jednotlivých případů tohoto ovládání.

Pro zobrazení menu souborů, resp. menu nastavení, je nutné táhnout prstem od levého, resp. pravého okraje směrem ke středu obrazovky. Při samotném pohybu bude vidět vyjíždějící menu, jak ukazuje obrázek 5.3. Zasunutí menu lze provést opačným pohybem.



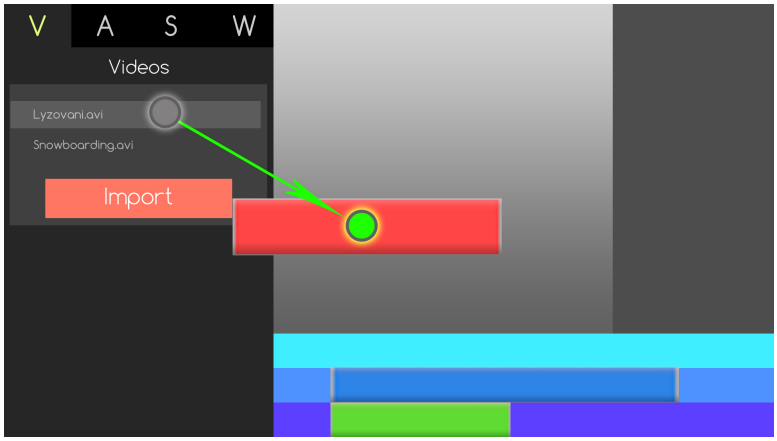
Obrázek 5.3: Ukázka vysunutí menu souborů.

Umístění importovaného souboru na časovou osu, je docíleno dlouhým stiskem daného importovaného souboru, přičemž se vytvoří nové médium, které je tahem umístěno na časovou osu. Po vytvoření tohoto média se zasune menu souborů, aby mohl uživatel umístit médium kdekoli na celou šířku časové osy. Tento případ naznačuje obrázek 5.4.

Přemísťování médií v rámci časové osy probíhá opět dlouhým přidržením média, stejně jako tomu bylo u importovaných souborů. V tu chvíli je médium vyjmuto z časové osy a čeká na nové umístění. Pokud médium přesuneme mimo časovou osu, bude médium odstraněno.

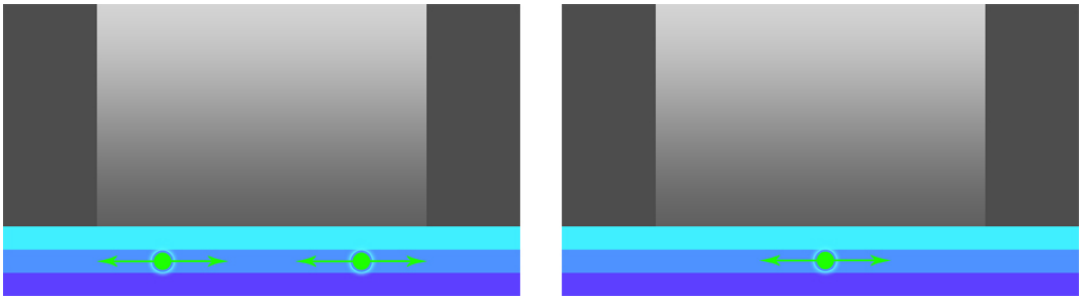
Ovládání komponenty časové osy probíhá rovněž pomocí gest. Konkrétně tahem dvěma prsty od sebe, nebo k sobě pro zvětšení nebo zmenšení zobrazovaného časového úseku, tahem jedním prstem do stran pro posun v čase zobrazovaného úseku, dlouhým stiskem pro přesun média a krátkým stiskem pro aktivaci daného média[4]. Na obrázku 7.3 je zobrazeno

[4]Aktivací média je zde myšlena akce, kdy je stisknuto médium a na základě jeho typu dojde ke změně obsahu menu nastavení a také obsahu *Actionbaru*.



Obrázek 5.4: Umístění importovaného souboru na timeline.

ovládání pomocí gest. První obrázek znázorňuje zvětšení a zmenšení zobrazované časové oblasti pomocí dvou prstů, druhý obrázek posun v čase.



Obrázek 5.5: Ovládání komponenty časová osa pomocí gest.

Kapitola 6

Implementace

Aplikace byla vyvíjena v operačním systému Ubuntu 14.04, ve vývojovém prostředí Eclipse Luna s ADT pluginem pro vývoj aplikací pro OS Android. Použitým programovacím jazykem byl jazyk Java 1.8.

Tato kapitola popisuje strukturu a některé zajímavé části této aplikace.

6.1 Struktura

Zdrojové soubory jsou rozděleny do několika balíků, přičemž kořenový balík *cz.janousek.videoeditor* obsahuje pouze soubor *Cons.java* obsahující konstanty. Názvy jeho podbalíků budou dále pro přehlednost uvedeny bez této kořenové cesty.

Balík *.activity* obsahuje všechny aktivity projektu, které byly popsány v kapitole 5.3, tudíž nebudou znova vysvětlovány. Dalším balíkem je *.db*, který obsahuje soubory pro práci s databází *DatabaseManager.java* a *DatabaseHelper.java*, které jsou vysvětleny v kapitole 6.4. Dále balík obsahuje třídy *Media.java*, *Project.java* a *ImportedFile.java*, jejíž instance jsou ukládány do databáze. Dalším balíkem je *.model*, který obsahuje mimo jiné dva důležité soubory *FrameModel.java* a *TimeLine.java*, které obsahují data pro třídy *TLView* a *FrameView*. Posledním balíkem je balík *.style*, obsahují třídy definující jednotlivé typy widgedů, které musejí implementovat rozhraní definované v souboru *Style.java*.

6.2 Styly

Styly jsou navrženy tak, aby bylo možné vytvářet nové styly a přidávat je do aplikace. Aby každý nový styl splňoval podmínky, které aplikace vyžaduje, musí implementovat rozhraní *Style*. Toto rozhraní deklaruje metody, které jsou využívány aplikací, prozatím zejména pro vykreslování widgedů (metody *drawWidget()*, *getWidth()*, *getHeight()*, a další). V dalším vývoji tohoto projektu bude nutné deklarovat nové metody do tohoto rozhraní podle potřeby. Každý ze stylů má také definovánu výšku a šířku svého widgedu, aby mohla být spočítána jeho pozice na plátně a tento widged nebyl vykreslován mimo plátno. Pozice je nastavená jako pozice středu widgedu vůči plátnu a metoda *drawWidget(Canvas canvas, int sensorValue)*, vykreslující tento widget na plátno, počítá vykreslování jeho částí relativně k tomuto středu. Díky tomuto způsobu vykreslování může být změněna pozice tohoto widgedu a ten je vždy vykreslován správně bez dalších úprav.

Vykreslení do bitmapy

Vykreslení do bitmapy, která je navrácena knihovnou *MediaLib*, probíhá následujícím způsobem:

1. Nejprve se vytvoří plátno z bitmapy pomocí konstruktoru *Canvas(Bitmap bitmap)*.
2. Poté je zavolána metoda *drawWidget(Canvas canvas, int sensorValue)*.
3. V této metodě je vykreslen widged na plátno pomocí standardních metod (*drawText()*, *drawLine()*, apod.) třídy *Canvas*, případně vykreslení *NinePatchDraweble* pomocí metody *NinePatchDrawable.draw(Canvas canvas)*.

6.3 Drawers

Na hlavní obrazovce aplikace (Main screen), se nachází dvě postranní menu (popsáno v kapitole 5.5). Tyto menu jsou implementovány pomocí *Android Navigation Drawer*. Implementace spočívá v definování XML struktury, kde je jako kořenový element *DrawerLayout* a postranní menu jsou elementy *RelativeLayout* s příslušným umístěním. Umístění elementů *RelativeLayout* je určeno pomocí atributu *android:layout_gravity* (Pro umístění na levé straně obrazovky je použita hodnota *start*, pro umístění napravo hodnota *end*).

Naplnění pravého menu nastavení probíhá dynamicky vytvořením nové instance třídy *View* pomocí metody *LayoutInflater.inflate(int resource, ViewGroup root)* a vložením tohoto view do elementu *FrameLayout*, který je dceřiným elementem elementu *RelativeLayout*.

6.4 Databáze

Implementace databáze spočívala ve vytvoření tříd *DatabaseHelper* a *DatabaseManager*. První jmenovaná třída slouží k nastavení a vytvoření databáze. Obsahuje jméno souboru *DATABASE_NAME*, ve kterém bude databáze ukládána a dále verzi této databáze *DATABASE_VERSION*. Pokud platí, že *DATABASE_VERSION* $= 1$ a současně databáze ještě není vytvořena, dojde k zavolání metody *onCreate()*, ve která proběhne vytvoření databáze. Pokud je *DATABASE_VERSION* větší než verze aktuální vytvořené databáze, dojde k volání metody *onUpgrade()*, ve které jsou definovány změny, podle nové verze databáze.

Třída *DatabaseManager* slouží pro ukládání, úpravu a mazání objektů v databázi. Pro třídu *Media* jsou zde implementovány metody *getAllMedia(Context c, Project p)*, *addMedia(Context c, Media v)*, *updateMedia(Context c, Media media)* a *deleteMedia(Context c, Media v)*. Tyto metody zajišťují všechny potřebné operace s tímto objektem. Obdobné metody jsou zde definovány i pro zbylé třídy *ImportedFiles* a *Project*, které jsou také ukládány do databáze.

6.5 Práce s knihovnou Ing. Tomáše Slavotínka

Aplikace využívá knihovnu Ing. Tomáše Slavotínka pro práci s video a audio soubory. Funkcionality není potřeba mnoho, přesto ne všechna potřebná funkcionalita je knihovnou poskytována. Pro využívání této knihovny je potřeba pracovat pouze se dvěma třídami. S třídou *MediaLib* a *MediaFile*. Třída *MediaLib* se využívá pro otevírání souborů a vytvoření instance třídy *MediaFile*. Dále bude tedy popisována práce zejména se třídou *MediaFile*, pokud nebude uvedeno jinak.

Knihovna definuje metody pro práci s video i audio sreamy souborů, nicméně po mnoha pokusech, byla úspěšně zprovozněna práce pouze s video streamy. Zda je knihovna nedokončená pro audio, nebo je v ní v tomto místě chyba a nebo zda nebylo vyzkoušeno její správné použití se nepodařilo zjistit. Vzhledem k absenci dokumentace a neúspěšných pokusech kontaktovat autora, by nejvhodnější další postup byl knihovnu přepracovat, nebo nahradit za jinou. Přesto, že je aplikace nachystána i pro práci s audio soubory, některé metody knihovny pro práci s audiem nefungují a proto se dále budeme věnovat pouze práci s videem.

Pro práci s videem jsou využívány tyto metody třídy *MediaFile*:

- *GetVideoFrameBitmap(Bitmap bitmap)* – sloužící pro získávání video snímku v interním čase souboru,
- *InsertVideoFrame(byte [] data)* – sloužící pro vložení video snímku do souboru,
- *SeekToTime(long pos, boolean bFast)* – sloužící pro posun interního času souboru,
- *Play()* – sloužící pro spuštění přehrávání souboru,
- *Pause()* – sloužící pro pozastavení přehrávání v souboru.

Použití

Při každém vytvoření média na časové ose[1] dojde k vytvoření instance třídy *MediaFile* a otevření souboru včetně potřebné inicializace. Tato instance je uložena v instanci třídy *Media*, která jej vytvořila. Je tedy vše nachystáno k práci se souborem.

Mód posunu

Pokud dojde k posunu komponenty *posuvník*, je zkoumáno zda je v aktuálním čase na časové ose nějaké médium. Pokud ano, je zavolána metoda *GetVideoFrameBitmap(Bitmap bitmap)* na příslušný *MediaFile* a bitmapa je vykreslena v komponentě *FrameView*.

Mód přehrávání

Pokud dojde ke spuštění přehrávání projektu je postup podobný jako v předchozí kapitole 6.5, jen je získávání snímku z videa cyklické. Při spuštění přehrávání se spustí přehrávání také přímo v knihovně a následně je cyklicky volána metoda *GetVideoFrameBitmap()*. Volání této metody ovšem také způsobuje posun v interním čase knihovny a proto je nutné tento čas zjišťovat a kontrolovat ihned po volání této metody pro případ, kdy by knihovna skočila na nepatrně odlišný čas, než byl očekávaný[2].

Export

Export videa probíhá stejným způsobem jako přehrávání, jen je nutná počáteční inicializace výstupního souboru. Výstupní soubor se vytváří postupným voláním funkcí:

- *OpenFile(String filePath, int nMode)* – pro vytvoření a otevření souboru,

[1] K vytvoření média na časové ose dojde po vložení importovaného souboru z levého menu na časovou osu, jako popsáno v kapitole 5.4

[2] K tomuto skoku v čase, ať už dopředu nebo dozadu, zřídkakdy dochází, zejména po předchozím posunu v čase.

- *InitVideoFrame(int width, int height, int format)* – pro inicializaci velikosti a formátu snímku,
- *InsertVideoStream(int nFrameWidth, int nFrameHeight, int nFPS, int nFormat)* – pro inicializaci video streamu,
- *InsertHeader()* – pro vložení hlavičky.

Dále se cyklicky získávají snímky ze zdrojových médií na časové ose a poté vkládají do výstupního souboru pomocí metody *InsertVideoFrame(byte [] data)*. Na konci dojde k uzavření všech souborů.

6.6 Časová osa

Jednou ze stěžejních částí aplikace je komponenta časová osa. Tato komponenta byla vytvořena jako *Custom View*[3], tedy třída, která dědí od třídy *View*. Veškeré další elementy, které se v této komponentě objevují jsou objekty přímo vykreslované na plátno, nikoli další instance třídy *View*, nebo jejich podtříd. Vzhledem k této odlišnosti od zbytku elementů, bude tato komponenta popsána podrobněji.

View

Pro tvorbu grafického uživatelského rozhraní nabízí Android několik základních prvků, které lze použít a dále upravovat. Mezi tyto elementy patří třídy, které dědí od *View* nebo třídy zděděné od *ViewGroup*. Nejpodstatnější rozdíl mezi těmito dvěma typy je, v možnosti vkládání dalších elementů přímo do těchto elementů.

Pro potřebu vytvoření časové osy, jakožto plochy, která se dá přibližovat, oddalovat, posouvat na strany a má obsahovat jakýsi seznam dalších elementů, by se nabízela možnost využít knihovní element *ListView*, do kterého by se přidávaly například elementy *ImageView*. Takovýto element *ListView* by pak musel být vytvořen pro každou vrstvu (audio, vide, senzor) a všechny dohromady by musely být v elementu *ScrollView*, kderý by umožňoval zvětšování a zmenšování podle potřeby.

Zde ovšem nastává problém s přibývajícími elementy, zastupujícími sekvence medii, které se umisťují do časové osy. Při velkém počtu těchto prvků, by byla uchovávána režie spojena s každým tímto prvkem, tj. instancí třídy, která dědí minimálně od třídy *View*. Proto byl zvolen způsob vytvoření jedné vlastní třídy zděděné od *View*, která bude zastřešovat celou časovou osu a vše ostatní bude pouze vykreslováno na plátno. V tom případě je zapotřebí si ukládat pouze režii spojenou s jednou instancí *View* a u každého prvku v časové ose pouze data spojena se samotnou reprezentací dat[3].

Časová osa je tedy implementována jako třída *TLView* rozšiřující třídu *View*. Mimo jiné metody obsahuje metodu *onDraw()*, která je volána systémem, při každém vykreslení, nebo po volání metody *invalidate()*. Metoda *onDraw()* vykresluje celý element časové osy, tj. časové měřítko, vrstvy a média v těchto vrstvách. Vykreslován je pouze aktuálně viditelný úsek časové osy. Toto řešení je důležité zejména pro projekty s velkým množstvím médií umístěných na časové ose.

Protože množina vykreslovaných elementů na časové ose není malá a metoda *onDraw()* by byla příliš složitá, jsou vytvořeny další třídy reprezentující jednotlivé části časové osy.

[3]Data uvedena ve schématu databáze u tabulky Média viz. 5.4.

Tyto třídy vlastní metodu *myDraw()*, která je volána z metody *onDraw()* třídy *TLView*. Každá z těchto tříd se tedy sama stará o své vykreslení. Jsou to třídy:

- *TLLineView* – reprezentující jednotlivé vrstvy,
- *TLTimeLineView* – reprezentující časovou osu,
- *TLItemView* – reperezentující média na časové ose.

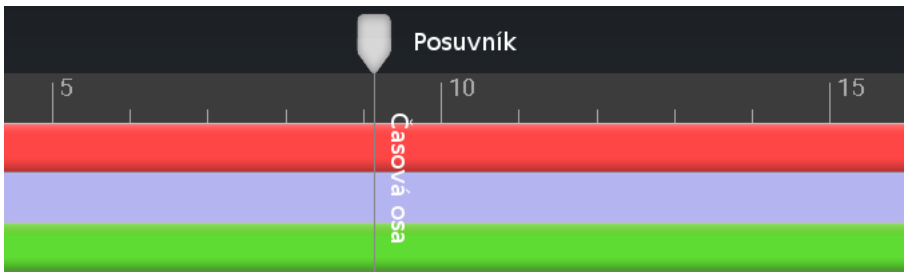
Data potřebná pro časovou osu, jako je pole objektů Média, pozice posuvníku, zvětšení, posunutí apod. jsou uložena ve třídě *TimeLine*.

Měřítko časové osy

Zajímavou částí z pohledu implementace je měřítko časové osy. Tato část časové osy je implementována ve třídě *TLTimeLineView* a vlastní dvě podstatné metody. Metodu *myDraw()*, vykreslující hodnoty a měřítko na plátno, a metodu *makeInterpolation()*. Druhá jmenovaná metoda naplňuje pole *timeValues* hodnotami, které mají být vykresleny. Algoritmus interpolace hodnot je založen na zjištění velikosti jednoho dílku stupnice a rozpočítání hodnot zobrazovaných na stupnici. Velikost tohoto dílku a také hodnoty jsou vybírány tak, aby byly pro uživatele co nejlépe čitelné. To znamená, že se snaží vybrat hodnoty vzdálené od sebe 5 s, 10 s, 15 s, 30 s, 1 min apod. Velikosti dílků jsou pak voleny v sekundách, desítkách sekund a podobně.

Posuvník

Posuvník je další vlastní vytvořenou komponentou, která se nachází přímo nad komponentou časová osa. *Posuvník* slouží k indikaci a úpravu času v projektu, přičemž zobrazení této konkrétní komponenty je pouze v úseku nad časovou osou, přesto, že by se mohlo zdát, že ji překrývá. Ve skutečnosti část (svislá čára), která je umístěna v časové ose je vykreslována touto komponentou, nikoli komponentou *Posuvník*. Komunikace mezi komponentou *Posuvník* a časovou osou probíhá nastavováním proměnné *seekPointerPos* třídy *Timeline*. Podle této proměnné je dále vykreslována samotná komponenta *Posuvník*, tak i svislá čára v komponentě časová osa. Obrázek 6.1 ukazuje rozdělení indikátoru posuvníku v těchto dvou popisovaných komponentách.



Obrázek 6.1: Ukázka vykreslování posuvníku.

Gesta

Pro ovládání této komponenty byly vytvořeny dva *listenery*[4]:

[4]Tzv. *listener* je technika, která je v OS Android využívána k definování reakce na určitou událost. Definovaná reakce je vykonána vždy, pokud dojde k této události.

- *SimpleOnScaleGestureListener* – pro detekci zvětšování/zmenšování pomocí dvou prstů,
- *SimpleOnGestureListener* – pro detekci posunu a dlouhého a krátkého dotyku.

Tyto listenery jsou volány v metodě onTouchEvent() třídy TLView. Jednotlivé metody listenerů definují události, které mají nastat. Popis ovládání této komponenty je popsán v kapitole 5.5.

Drag and Drop

Drag and drop je v systému OS Android framework, díky kterému je možné přesouvat data z jednoho objektu View do druhého. V této aplikaci je tento framework využit ve dvou případech. Jedná se o:

- přesun importovaného souboru na časovou osu
- a přesun Médií v rámci časové osy, popř. jejich odstranění.

Pro využití tohoto frameworku je nutné nejprve vytvořit objekt typu *ClipData* a poté implementovat *OnDragListener*, který je nastaven pomocí metody *setOnDragListener()* objektu, který má být cílovým objektem přesunu[5].

[5]Cílový objekt musí být opět instancí třídy View nebo instancí její podtřídy

Kapitola 7

Testování a zhodnocení

Testování

Testování probíhalo v průběhu vývoje aplikace na fyzickém zařízení Nexus 7 s OS Android verze 5.0. Tento přístroj má úhlopříčku displeje 7 palců s rozlišením 800 × 1280 px. Přístroj je poháněn čtyřjádrovým procesorem Cortex-A9 o taktu 1.2 GHz[1].

Na zařízení byla testována rychlost exportu videa z projektu. Jednalo se o export samotného videa, videa s widgedem Snow a videa s widgedem Skydiving. Výstupem pro každou z možností byly 3 videa ve formátu mp4 s rozlišeními 640 × 360, 960 × 480 a 1280 × 720 px. Čas byl měřen ručně v 10 pokusech a výsledný čas byl zprůměrován a zaokrouhlen na sekundy. Dosáhlo se následujících výsledků:

Rozlišení	640 × 360	960 × 480	1280 × 720
Bez widgedu	0:41	1:40	2:31
Widged Snow	0:57	2:04	3:31
Widged Skydiving	1:37	3:23	5:38

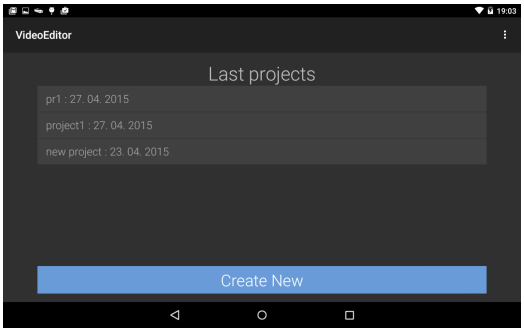
Zajímavým výsledkem je časový rozdíl mezi oběma widgedy, který je způsoben jejich navržením. Zatímco widged Skydiving je tvořen tzv. 9-patch souborem a jednou ručně vykreslovanou čarou, widged Skydiving je celý složen z čar vykreslovaných přímo na plátno (viz. kapitola 6.2). Z tohoto výsledku lze odvodit, že pro budoucí návrh a implementaci widgedů je výhodnější použít 9-patch soubor pro části widgedu, které jsou neměnné.

Screen shots

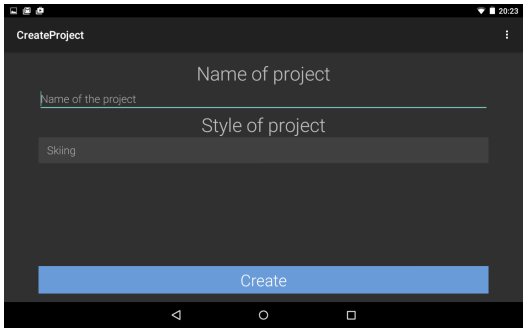
V této kapitole jsou zobrazeny reálné screenshoty aplikace, pořízené přímo na fyzickém přístroji Nexus 7. Na prvních snímcích 7.1 je ukázka přehledu vytvořených projektů a vytvoření nového projektu.

Na dalších snímcích jde vidět práce s hlavní obrazovkou, konkrétně operace přesunu importovaného souboru na komponentu časová osa.

Na posledním snímku je opět ukázka hlavní obrazovky se zobrazeným snímkem v aktuálním čase a vykresleným widgedem Skydiving.

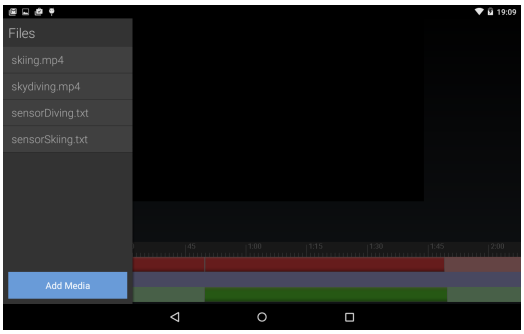


(a) Projects Screen.

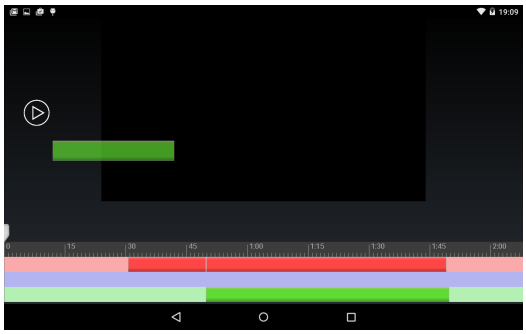


(b) New Project Screen.

Obrázek 7.1: Ukázka aplikace.

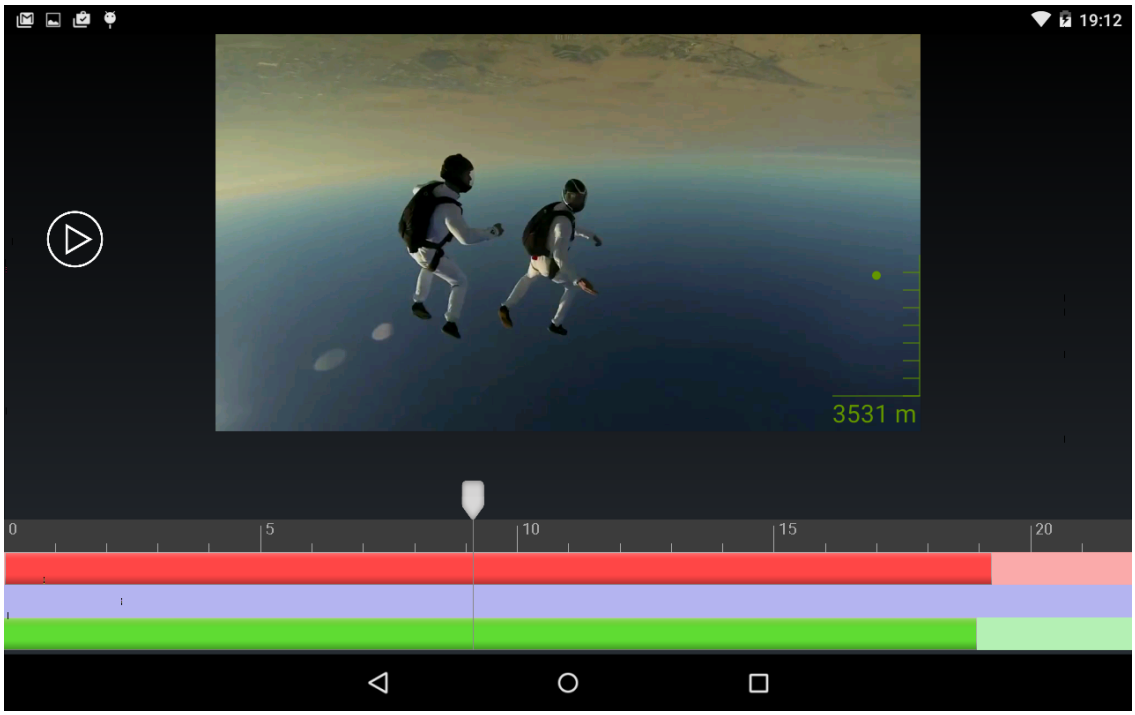


(a) Před začátkem přesunu.



(b) V průběhu přesunu.

Obrázek 7.2: Main Screen: Drag and drop



Obrázek 7.3: Main screen: Přesouvání v čase.

Kapitola 8

Závěr

Úkolem bylo vytvořit video editor pro střih sportovních videí na platformě android. Součástí řešení je i skript pro vytvoření souboru s pseudo daty imitující data ze senzorů. Editor dokáže pracovat i s více videi současně. Uživatel má možnost videa stříhat a různě za sebe skládat na časovou osu. K videům je také možnost připojení souboru s daty ze senzorů a tato data vykreslit jako jeden ze dvou typů předdefinovaných widgedů. Aplikace dokáže zobrazovat náhled projektu v zadaném čase, přehrát jej a vyexportovat do předdefinovaných formátů. Dále je možné vytvořené projekty ukládat do databáze SQLite pro další úpravu.

Během vytváření aplikace nastalo mimo jiné několik problémů s knihovnou pro zpracování videa, které nebylo možné kvůli chybějící funkčnosti vyřešit. V některých případech bylo možné tyto problémy obejít, nicméně v některých případech nikoli. Jedním z případů je nefunkčnost při práci s audio soubory.

Z hlediska dalšího postupu při vývoji aplikace by tedy bylo vhodné vyměnit knihovnu pro práci s video a audio soubory. Dalším krokem, vyjma zprovoznění nové knihovny a s ní i práce s audiem, by byla jednak ve vývoji nástrojů, pro úpravu jednotlivých médií a jednak v rozšíření možností stylů. Co se týká nástrojů, by byla vhodná například implementace barevných filtrů na video soubory nebo efekt zpomalení. Co se týká rozšíření možností stylů, by bylo vhodné doplnit styly například o přednastavení audio souboru vhodného žánru nebo definici přechodů mezi jednotlivými video soubory. Dále by bylo vhodné vytvoření například textových widgedů a celkově možnost umístění kamkoli na plátno.

Možností, jak lze aplikaci dále vyvíjet a rozšiřovat je mnoho, nicméně původním záměrem bylo dosáhnout automatického střihu videa, a proto se další vývoj bude zaměřovat především tímto směrem.

Literatura

[1] GSM arena: Asus Google Nexus 7 (2013) [online]. http://www.gsmarena.com/asus_google_nexus_7_%282013%29-5600.php, 2013 [cit. 2015-5-4].

[2] Android Developers: Activity [online]. http://developer.android.com/reference/android/app/Activity.html, 2015 [cit. 2015-5-4].

[3] Android Developers: Custom Components [online]. http://developer.android.com/guide/topics/ui/custom-components.html, 2015 [cit. 2015-5-4].

[4] Android Developers: Dashboards [online]. https://developer.android.com/about/dashboards/index.html, 2015 [cit. 2015-5-4].

[5] Android Developers: Media [online]. http://developer.android.com/reference/android/media/package-summary.html, 2015 [cit 2015-5-4].

[6] Android Developers [online]. https://developer.android.com/guide/index.html, 2015 [cit. 2015-5-4].

[7] Android [online]. http://www.android.com/, 2015 [cit. 2015-5-4].

[8] FFmpeg [online]. https://www.ffmpeg.org/, 2015 [cit. 2015-5-4].

[9] Google Play [online]. https://play.google.com/store, 2015 [cit. 2015-5-4].

[10] JavaCV [online]. https://github.com/bytedeco/javacv, 2015 [cit. 2015-5-4].

[11] JJmpeg [online]. http://jjmpeg.sourceforge.net/, 2015 [cit 2015-5-4].

[12] Open handset alliance [online]. http://www.openhandsetalliance.com/, 2015 [cit. 2015-5-4].

[13] Statista [online]. http://www.statista.com/statistics/266572/market-share-held-by-smartphone-platforms-in-the-united-states/, 2015 [cit. 2015-5-4].

[14] Allen, G.: *Android 4*. Brno: Computer Press, první vydání, 2013, ISBN 978-80-251-3782-6.

[15] Aydin, M.: *Android 4*. Birmingham, UK: Packt Pub., třetí vydání, 2012, ISBN 978-1-84951-952-6.

[16] Burnette, E.: *Hello, Android*. Raleigh, N.C.: Pragmatic Bookshelf, třetí vydání, 2010, ISBN 19-343-5656-5.

[17] Marko Vitas: ART vs Dalvik - introducing the new Android runtime in KitKat [online]. `https://www.infinum.co/the-capsized-eight/articles/art-vs-dalvik-introducing-the-new-android-runtime-in-kit-kat`, 2013 [cit. 2015-5-4].

[18] Slavotínek, T.: *Knihovna pro práci s videem pro platformu Android*. Diplomová práce, FIT VUT v Brně, 2014.

[19] Verduzco, W.: XDA developers [online]. `http://www.xda-developers.com/android-5-0-lollipop-changelog/`, 2014 [cit. 2015-5-4].

Příloha A

Obsah DVD

Součástí práce je DVD se zdrojovými kódy aplikace a dalších souborů potřebných k jejímu spuštění. Na DVD se rovněž nachází zdrojové kódy této zprávy a vytvořený plakát.

- `Source` – tato obsahuje zdrojové soubory aplikace, včetně souboru `README` s instrukcemi pro spuštění. Dále obsahuje vygenerovaný *.apk* soubor pro instalaci přímo na mobilní zařízení s OS Android a skript, který byl využitý pro generování pseudo dat senzorů.
- `Poster` – tato složka obsahuje plakát v plném rozlišení i ve zmenšené verzi ve formátu PDF.
- `FilesToImport` – tato složka obsahuje demonstrační soubory využité aplikací (videa a data ze senzorů).
- `Report` – tato složka obsahuje zdrojový text tohoto dokumentu, včetně vygenerovaného PDF dokumentu.

Příloha B

Plakát

Obrázek B.1: Demonstrační plakát aplikace. [6]