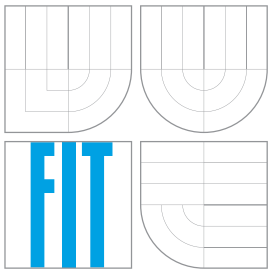
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

NÁSTROJ PRO PREZENTOVÁNÍ VIDEÍ A DALŠÍHO OBSAHU NA OBRAZOVCE S DOTYKOVÝM VSTUPEM

A TOOL FOR PRESENTING VIDEOS AND OTHER CONTENT ON A TOUCHSCREEN

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LIBOR ZAPLETAL

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. ADAM HEROUT, Ph.D.

BRNO 2013

Abstrakt

Práce popisuje vývoj aplikace pro systémy Windows pomocí technologie WPF zaměřené na prezentování videí s využitím dotykového ovládání. Aplikace je velice jednoduchá na ovládání a jejím hlavním účelem je usnadnit uživateli výběr obsahu, který ho zajímá. Práce popisuje hlavní rysy technologie WPF, analýzu konkurečních řešení, samotný návrh vlastní aplikace a její implementaci. Dále tato práce popisuje řešení různých problémů při vývoji. Na konci práce je zhodnoceno výsledné řešení, jeho další vývoj a také způsob testování.

Abstract

This bachelor thesis describes the development of application for Windows using WPF focused on presenting videos using touch. The application is very easy to use and its main purpose is to facilitate the users to select the content that interests them. This thesis describes the main features of WPF, analysis of competing solutions, application design and its implementation. Furthermore, this thesis describes various problems during development. At the end of the thesis are evaluated the resultings of solution, its further development and testing method.

Klíčová slova

Video, multimediální obsah, multimédia, prezentace, přehrávač, náhledy, obrazovka s dotykový vstupem, dotyk, WPF, C#, Windows, XAML

Keywords

Video, multimedia content, multimedia, presentation, player, previews, touchscreen, touch, WPF, C#, Windows, XAML

Citace

Libor Zapletal: Nástroj pro prezentování videí a dalšího obsahu na obrazovce s dotykovým vstupem, bakalářská práce, Brno, FIT VUT v Brně, 2013

Nástroj pro prezentování videí a dalšího obsahu na obrazovce s dotykovým vstupem

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing. Adama Herouta, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Libor Zapletal
14. května 2013

Poděkování

Na tomto místě bych rád poděkoval doc. Ing. Adamovi Heroutovi za věnovanný čas a také za cenné připomínky a odborné rady, kterými přispěl k vypracování této práce.

Obsah

Kapitola 1

Úvod

V poslední době se rozmohla dotyková zařízení a stala se cenově dostupnými. Hlavně se jedná o chytré mobilní telefony a tablety, ale nejsou to jediná zařízení. Díky rostoucímu zájmu a klesající ceně se stávají dostupnými i větší dotyková zařízení jako například monitory, které si najdou velké uplatnění v různých oborech a službách.

Cílem této bakalářské práce je vytvořit aplikaci, které by právě na zmíněných monitorech dokázal prezentovat video obsah pro náhodné kolemjdoucí. Tato aplikace by měla mít široké využití bez nutnosti předpokládat nějaké zkušenosti uživatelů s ovládáním dotykových zařízení. Uživatel by měl mít možnost snadno si vybrat video z nabídky a to podle různých paramatrů.

Samotný text této práce je rozdělen do několika kapitol. Nejdříve tedy popisuji současná řešení a případnou konkurenci své aplikace v kapitole 1. Poté jsem si vybral jako technologii pro implementaci WPF, které popisuji v kapitole 2. V kapitole 3 je popsán návrh výsledné aplikace a v kapitole 4 je popsána samotná implementace aplikace. Kapitola 5 se zabývá vyhodnocením výsledného řešení a jeho testováním. Poslední kapitola shrnuje celý vývoj aplikace, možnosti dalšího rozšíření aplikace a její další vývoj.

Kapitola 2

Analýza existujících řešení

V této kapitole bych rád popsal existující řešení, která mě inspirovala k výsledné aplikaci. Některá z těchto řešení jsem měl možnost sám si vyzkoušet a jiná se mi podařilo dohledat. Díky získaným zkušenostem z těchto zařízení jsem pak vycházel při definování požadavků a návrhu vlastního řešení. Z těchto poznatků jsem pak vycházel při výběru technologie pro implementaci řešení, kterou popisuji v následující kapitole.

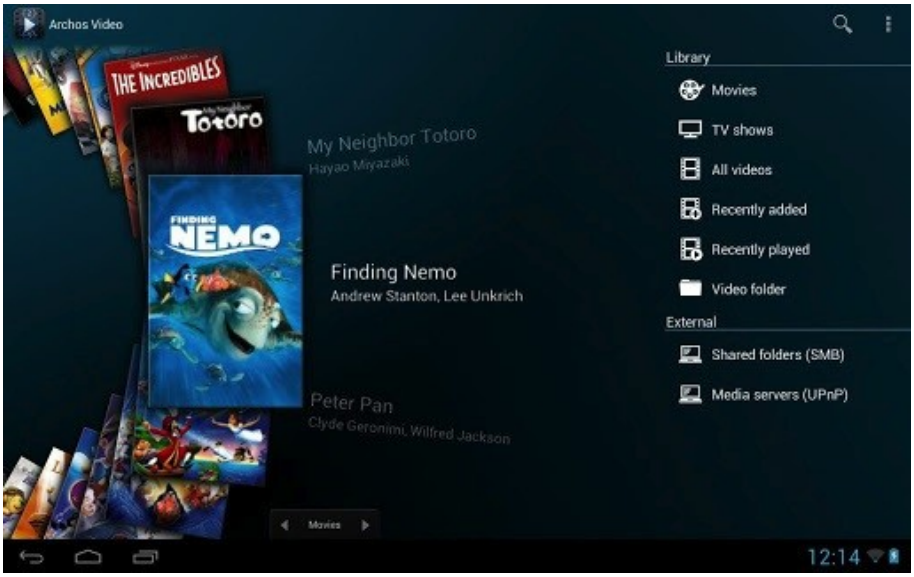
Pokud ovšem jde o nějakou přímou konkurenci, tak tu se mi bohužel dohledat nepodařilo. Hledal jsem hlavně aplikaci pro přehrávání multimediálního obsahu na velkém dotykovém zařízení a žádnou podobnou aplikaci se mi nepodařilo najít. Buďto šlo o komplexní aplikace, kdy přehrávání videa bylo jednou z mnoha funkcí (a ne moc podstatnou) a nebo naopak šlo o neinteraktivní aplikaci, kde se postupně přehrála všechna videa (případně jedno ve smyčce).

Nejčastější zařízení s podobnou funkcí jsou kiosková řešení, která jsou vyráběná a dodávaná přímo na míru firmám, které si sami nechají vytvořit na zakázku pro svoje potřeby. Toto řešení je spojováno i s výrobou samotného zobrazovacího zařízení, kdy je vytvořeno v designu dané společnosti (či produktu), kterou má propagovat. U tohoto řešení se nejčastěji využívají hlavně menší dotykové displeje (o velikosti od $10''$ do $18''$). Vzhledem k specializaci a zakázkové výrobě takových zařízení je jejich výsledná cena je poměrně velká a proto bych chtěl se svou aplikací nabídnou řešení pro zákazníky, kterým bude vyhovovat levnější řešení pro jakýkoliv dotykový monitor bez nutnosti nějakého speciálního zařízení.

2.1 Aktuální řešení v CVT na FIT VUT

Jednou z inspirací při vývoji pro mě bylo aktuální řešení na chodbě v CVT na Fakultě Informačních technologií VUT. Aktuálně je na zobrazovacím zařízení nainstalována Linuxová distribuce Ubuntu a spuštěn skript, který prochází určitou složku a přehrává videa z této složky. Ač toto řešení není interaktivní, tak mi pomohlo specifikovat požadavky, které jsem na výslednou aplikaci měl:

- Jednoduchá aplikace, kterou stačí spustit a dál není potřeba se o nic starat
- Videa stačí přidat do složky a automaticky se přidají do aplikace
- Pokud se s aplikací nic neděje, tak začít postupně přehrávat všechna videa
- Pokud je video přehráváno, tak ukázat uživateli, že může aplikaci ovládat



Obrázek 2.1: Pěkný výběr z filmů v knihovně v přehrávači Archos Video Player pro Android

- Zákazat uživateli aplikaci vypnout nebo ovládat něco jiného než danou aplikaci

Dále pak mně dané řešení pomohlo i při představování si, jak různá videa mohou být, co a kde se v nich může vyskytovat (například něco důležitého v levém horním rohu) a že i s tím bych měl počítat a to například při návrhu ovládacích prvků. Ptal jsem se také uživatelů, jakou funkčnost by mé řešení mělo obsahovat a velice mi pomohlo, když jsem mohl odkazovat na stávající řešení, kdy mé řešení by mělo mít podobný účel, ale právě díky dotykovému ovládání by si uživatel mohl vybrat, která videa by chtěl přehrávat.

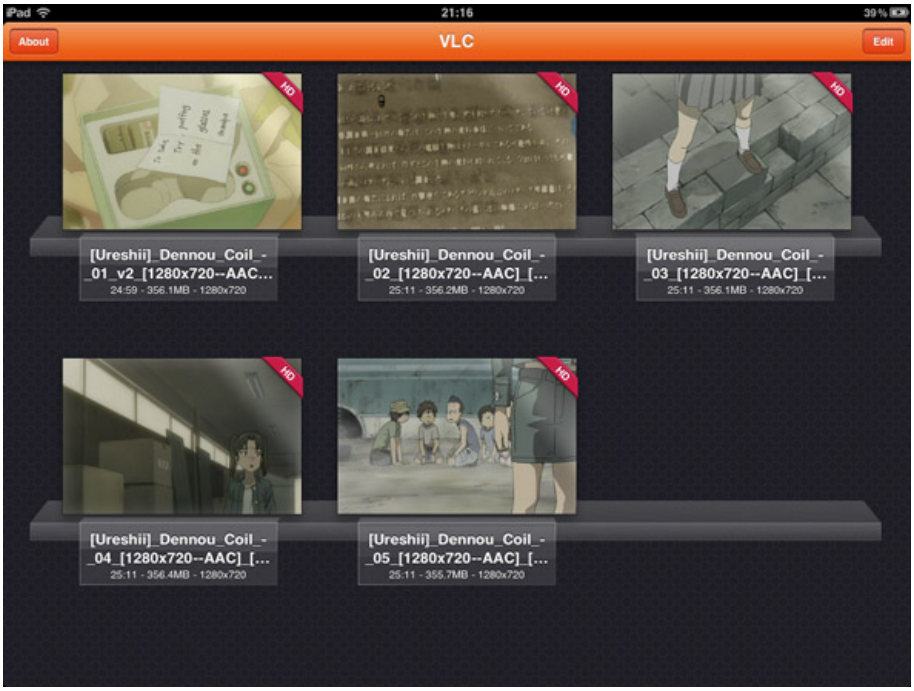
2.2 Popis některých konkurečních aplikací

Jak jsem se zmínil, tak přímou konkurenci jsem pro svoji aplikaci nenašel a proto jsem se zaměřil více na aplikace s podobným účel. Tedy hlavně na aplikace přehrávající multimediální soubory na dotykových zařízeních a vzhledem k tomu, že mobilní zařízení mají svá specifikace (díky menší velikosti obrazovky), tak jsem se spíše díval na aplikace, které jsou dostupné pro tablety.

Většinou vlastností a funkcí se programy pro přehrávání videí neliší a mnohé Vám prostě nějakým způsobem zpřístupní seznam dostupných videí (ať již v nějaké multimediální knihovně nebo z dané složky či disku) a vy si můžete vybrat, které video chcete přehrát. Mezi těmito přehrávači graficky vyčníval Archos Video Player pro Android, který se ovšem zaměřuje hlavně na filmy a tv seriály a doplňuje tedy potřebné informace (obrázky, popisy apod.) z internetu z dostupných databází, což není můj cíl. Tady jsem si specifikoval, že moje řešení musí být co nejobecnější a tedy nepředpokládám žádné dohledatelné informace k obsahu videa. Jediné informace, které mohu k daném obsahu získat budou od autorů daného videa, ale ani ty nesmí být podmínkou.

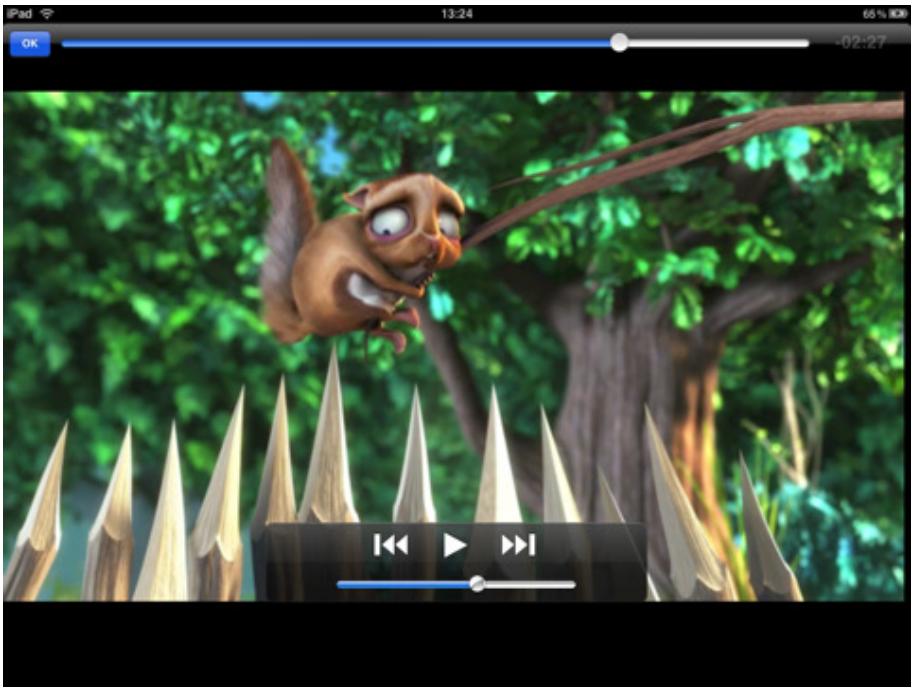
Zaměřil jsem se tedy na ostatní přehrávače, jestli by se u nich nedala najít nějaká inspirace a například pro návrh menu mě zaujala možnost výběru videí u přehrávače VLC media player pro iPad. Bývá to jedna z častých možností, jak si vybírat videa. Náhledy

vypadají hezky a člověk už z nich vidí, co by mohl ve videu očekávat. Naopak dojem kazí popisky videa, které jsou brány z názvů souborů a tomuto se chci u svého řešení vyhnout.



Obrázek 2.2: Ukázka výběru videí ve VLC media playeru na iPadu

Podobně se lze inspirovat i u jednoduchosti ovládání přehrávače při přehrávaném videu a je potřeba zapřemýšlet, které funkce jsou v mém řešení potřeba a které ne.



Obrázek 2.3: Ovládací prvky u přehrávaného videa

Kapitola 3

Windows Presentation Foundation

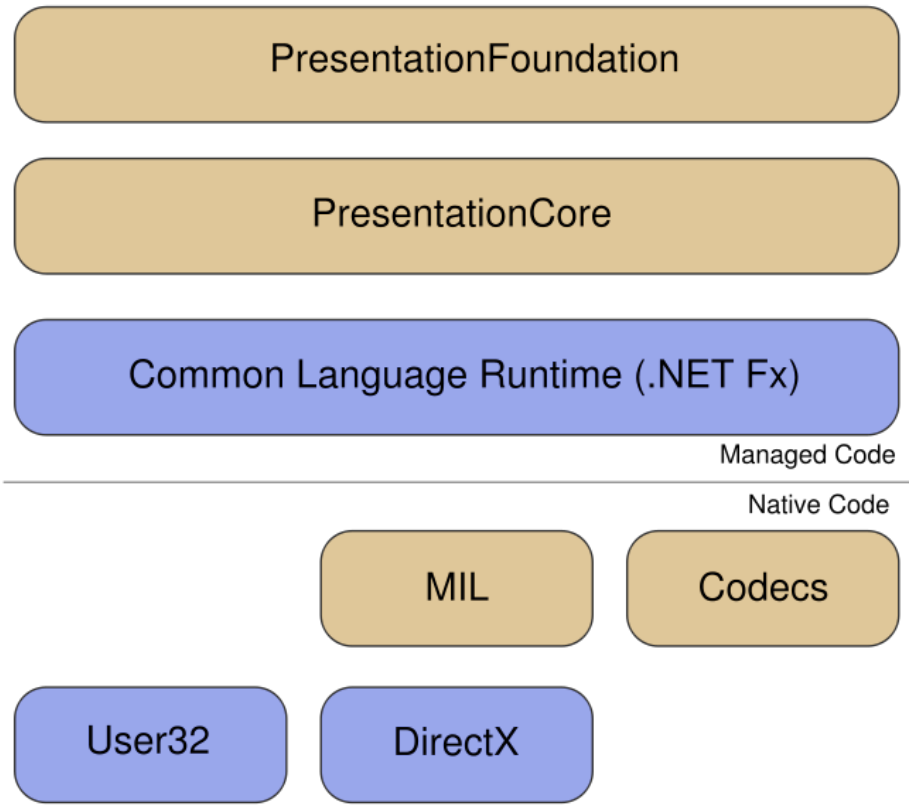
Windows Presentation Foundation (dále jen WPF) je grafický systém sloužící k vykreslování uživatelských rozhraní v aplikacích pro operační systémy Windows. WPF, dříve známé jako Avalon, bylo poprvé vydáno jako část .NET Frameworku verze 3.0. WPF využívá DirectX k vykreslování bohatých uživatelských rozhraní (Rich User Interface - RUI) místo staršího GDI rozhraní. WPF se snaží o poskytnutí jednotného programovacího modelu pro vytváření aplikací a odděluje uživatelské rozhraní od business logiky. WPF je nástupce starší technologie Windows Forms (označované také jako WinForms), která se ovšem může ještě stále využívat. WPF aplikace může být nasazena buďto jako samostatná desktopová aplikace nebo jako aplikace běžící na vzdáleném serveru (která má ovšem k dispozici daleko větší možnosti než běžná webová aplikace).

WPF používá XAML (jazyk založený na XML syntaxi) k definování uživatelského rozhraní (User Interface - UI), o kterém se zmíním v další kapitole. WPF se snaží o zjednodušení a poskytnutí více možností vývojáři, pro lepší a bohatší UI. WinForms pracují s elementy uživatelského prostředí vestavěné přímo v systému, které není možné snadno upravovat, ať už po stránce vzhledové nebo po stránce chování, protože při navrhování WinForms se s tím zkrátka nepočítalo. WPF je už od začátku navrhováno s důrazem na volnost při navrhování vzhledu a je tedy možné si navrhnout vlastní ovládací prvky s různým chováním a reakcemi podle uživatelova ovládání. Ovšem je možné také využít základní prvky, které jsou již definované a použít ve své aplikaci i tyto prvky (podobně jako WinForms).

V současné době je aktuální verze WPF 4.5 ze srpna 2012, což je již 5. hlavní verze WPF vydaná od roku 2006, kdy poprvé bylo vydáno WPF (ve verzi 3.0, označováno podle verzí .NET Frameworku, které WPF obsahuje). WPF je součástí všech systému Windows od systému Vista a do starších systémů jako Windows XP SP2/SP3 je možné doinstalovat potřebné knihovny a také spouštět WPF aplikace.

3.1 Architektura WPF

Architektura WPF[10] byla navržena s cílem poskytnout komplexní možnosti v návrhu UI při co nejvyšším možném výkonu a stále zachovat pohodlnost a snadnost práce. Architekturu je možné vidět na obrázku 3.1. Jak je vidět, tak architektura WPF využívá komponenty, jak z managed kódu, tak i z nativního kódu. Managed kód[1] je kód, který je řízen běhovým prostředím, které zajišťuje o základní služby jako je automatická správa paměti, typová verifikace apod. Managed kód z WPF využívá běhové prostředí Common Language Runtime (dále CLR). Více o architektuře .NET v na MSDN v sérii článků věnujících se



Obrázek 3.1: Obrázek architektury WPF

této technologii [5].

Ačkoliv tedy technologie WPF využívá obou možností, tak veřejné API, které využíváme při programování WPF aplikace (např. v jazyku C#) je dostupné jen skrz managed kód. Celá architektura WPF je postavena nad DirectX technologií, která v systémech Windows slouží pro přímou komunikaci s hardwarem (nejčastěji grafickou kartou). Ke komunikaci s DirectX se používá Media Integration Layer (MIL), což je engine zajišťující vykreslení WPF aplikace, která je implementován jako nativní komponenta a je obsažen v souboru `milcore.ddl`. Kromě MIL jsou ještě v nativním kódu implementovány kodeky, které jsou součástí souboru `windowscodecs.ddl`. S vrstvou MIL a Codecs komunikuje .NET assembly PresentationCore, který již běží v rámci CLR (zmíněného výše) a nabízí to nejzákladnější rozhraní WPF, se kterým můžeme pracovat. Úplně na vrchu sedí assembly PresentationFramework. Rozšiřuje PresentationCore o komplexní prvky uživatelského prostředí a pokročilejší funkce.

3.2 Značkovací jazyk XAML

Extensible Application Markup Language neboli XAML je značkovací jazyk vyvinutý společností Microsoft pro popis grafického rozhraní u technologií od Microsoftu. Nejdříve byl využit u WPF, ale postupem času Microsoft implementoval tento jazyk i u dalších technologií jako Silverlight, Windows Workflow Foundation, Windows Phone a Windows RT. U WPF slouží XAML k definování elementů UI, data bindingu, přiřazování událostí a mnoha dalším

```
<Window x:Class="test.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <StackPanel Orientation="Horizontal" VerticalAlignment="Top">
        <TextBox Text="Zadejte Vasi zpravu" />
        <Button Content="Odeslat" />
    </StackPanel>
</Window>
```

Obrázek 3.2: Ukázka XAML kódu

funkcím.

Princip provázanosti XAMLu a WPF je jednoduchý. Jednotlivé elementy v XAMLu jsou mapovány přímo na instance objektů z CLR a atributy XAML elementů jsou zase mapovány na CLR vlastnosti a události těchto objektů.

Soubory XAMLu je možné vytvářet a upravovat pomocí nástrojů jako jsou Microsoft Visual Studio (dále VS) a nebo Microsoft Expression Blend. Samozřejmostí je také možnost upravit tyto soubory i v textových editorech, ovšem už bez podpory náhledu vzhledu jednotlivých souborů. Všechno co lze vytvořit nebo implementovat pomocí XAMLu, lze také vytvořit v .NET jazyku jako je C# nebo Visual Basic.NET. Ovšem právě jedním z důležitých aspektů, proč byl jazyk XAML vytvořen, je usnadnění oddělení návrhu vzhledu od funkčnosti aplikace a tím i umožnit vytvářet obě části nezávisle na sobě.

Tento jazyk je velice oblíbený pro popis UI a to pro jeho jednoduchost a srozumitelnost i bez předchozích znalostí. Jazyk XAML popisuje UI jako strom jednotlivých prvků pomočí rozšířené XML syntaxe. Kromě velkého množství základní elementů, nad kterými je možné snadno si vytvořit vlastní, tak má XAML výbornou podporu pro flexibilní rozvržení jednotlivých prvků na obrazovce a to díky několika základním elementům. Mezi ty nejčastěji používáné patří Grid, StackPanel, WrapPanel, Canvas a mnoho dalších. Díky těmto prvkům je poté možné navrhovat aplikaci, tak aby i při změně velikosti okna se všechny prvky správně roztáhly a posunuly.

http://msdn.microsoft.com/en-us/library/ms752347.aspx

3.3 Výhody a nevýhody WPF

Výhody:

- XAML
  Velkou výhodou je již zmiňovaný značkovací jazyk. Kromě zmiňovaných vlastností jazyku v sekci věnující se XAMLu je možné také ještě navíc zmínit výborné provázání s daty z kódu. Jedná se o tzv. Data Binding[6].

- Nenáročnost
  Díky využití DirectX místo GDI je renderování aplikace přenecháno grafické kartě, která tím značně ulehčuje práci procesoru počítače a tím se aplikace stává rychlejší a svižnější i za předpokladu RUI.

- Jazyky C# a Visual Basic

Funkční logiku aplikace ve WPF je možné vytvářet pomocí jazyků C# a Visual Basic, což jsou vysokoúrovňové objektově orientované programovací jazyky. Já osobně jsem si pro svůj projekt vybral jazyk C#, protože vychází z jazyků Java a C++, které jsou mi blízké a pro jeho značnou oblíbenost mezi vývojaři, která vychází z jeho vlastností. Jazyk C# je aktuálně ve verzi 5.0 a s každou další verzí Microsoft rozšiřuje možnosti jazyka.

Nevýhody:

- Omezení na OS Windows
  Vzhledem k tomu, že WPF je technologie od Microsoftu, tak i podle toho vypadá kompatibilita s operačními systémy, kdy jsou podporovány pouze systémy Windows a to od verze XP SP2. Aktuálně neexistuje žádná jednoduchá portace aplikace pro ostatní OS jako například Linux. Ani skupina Mono, která se specializuje na port technologií od Microsoftu, nemá v blízké době zájem o předělání WPF[1].

3.4 Proč jsem vybral WPF pro řešení

Jakmile jsem zanalyzoval existující řešení a měl představu o tom, co by měla moje aplikace umět a kde by se měla využívat, tak jsem začal uvažovat nad tím, pomocí jaké technologie bych ji chtěl implementovat. Uvažoval jsem nad následujícími technologiemi:

- Java společně se nějakou knihovnou pro GUI (AWT, Swing, Qt, ...)
- `C++` a MFC nebo Qt
- C# a WPF

Nakonec jsem si vybral třetí řešení právě pro již zmíněné výhody a hlavně tedy pro velké možnosti při návrhu GUI. Vybral jsem si jako OS systém pro implementaci své aplikace Windows 7 pro jeho rozšířenost a oblíbenost a zavrhnul jsem tedy multiplatformní řešení. Což mi ovšem na druhou stranu dávalo výhodu právě ve využití WPF a jeho možnost při vytváření uživatelského rozhrání.

3.5 Budoucnost WPF

Jak již bylo zmíněno výše, tak WPF je technologie od Microsoftu a její budoucnost je spjatá s budoucností operačních systémů Windows, které aktuálně jsou velice rozšířené a oblíbené a proto i aplikace vytvářené ve WPF mají k dispozici velký trh, na kterém mohou být využívány.

Pokud jde o aktualizace, tak Microsoft vydává nové verze .NET Frameworku společně s novými verzemi svých OS a s tím i vycházejí nové verze technologii WPF, které přinášení moderní technologie a nové možnosti pro WPF. Poslední verzi OS Windows 8 vydal i ve verzi pro procesory ARM, která je označována jako Windows 8 RT[2] a která již neobsahuje možnost spustit WPF aplikace (ani WinForms aplikace) a tato verze je určena hlavně pro tablety. Ovšem právě kvůli velkému omezení je tato verze neoblíbená a výrobci tabletů raději používají „obyčejná“ Windows 8, na kterých WPF aplikace v pořádku fungují. S Windows

[1] Více o Mono a jejich portaci WPF na http://www.mono-project.com/WPF
[2] Více o Windows RT na http://en.wikipedia.org/wiki/Windows_RT

8 přišla nová verze .NET Frameworku a to ve verzi 4.5, která kromě spousty novinek pro ostatních technologie, přinesla další rozšíření vlastností WPF.

A právě proto si myslím, že prozatím má technologie WPF budoucnost a je to jedna z nejzajímavějších a nejdůležitějších technologií, pokud chcete vyvíjet aplikace pro operační systémy Windows.

Kapitola 4

Návrh aplikace

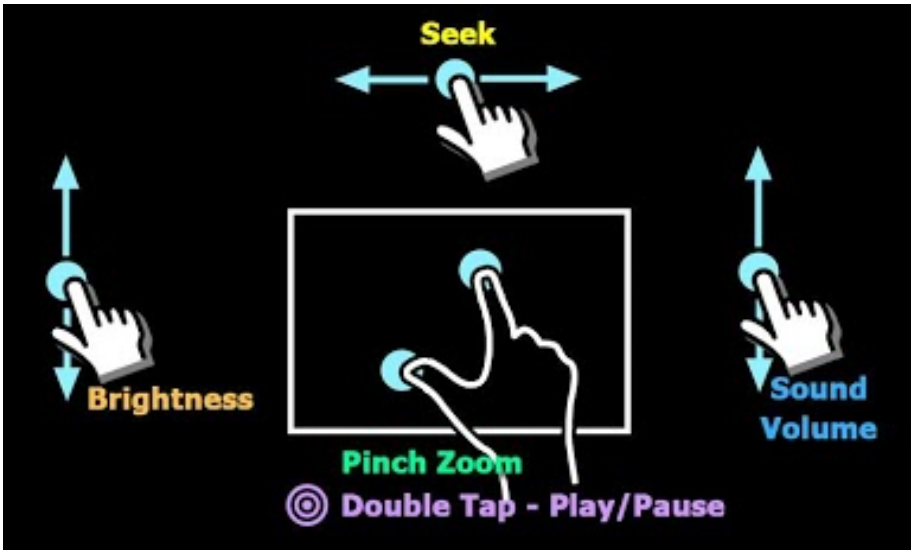
Poté co jsem si prohlédnul aplikace s podobným zaměřením a vybral jsem si technologii pro implementaci, tak jsem začal navrhovat svou aplikaci. Nejdříve jsem se tedy rozhodl, že aplikace by měla obsahovat dva druhy zobrazení, mezi kterými by se velice snadno přepínalo. První zobrazení by mělo být menu, kde by se dalo velice snadno vybírat z dostupného obsahu potřebná videa. Druhé zobrazení by přehrávalo daný obsah společně s ovládacími prvky, které by umožňovaly uživateli snadno ovládat samotné přehrávání videa. Dále bylo potřeba zaměřit se na cílové zařízení a s tím navrhovat jednotlivá zobrazení (velikost prvků, umístění apod.).

4.1 Návrh aplikace pro větší dotyková zařízení

Pokud se podíváme na některou z příruček pro různé platformy s dotykovým ovládáním (Android příručka[2], iOS příručka[3]), tak zjistíme, že jednou z důležitých vlastností pro návrh dotykové aplikace je využití různých dotykových gest. Tomuto jsem se chtěl u své aplikace vyhnout a to hlavně proto, že aplikace by neměla předpokládat žádné znalosti uživatele s dotykovými obrazovkami a s nějakými specifickými gesty a měla by být co nejjednoduší, aby uživatel vždy věděl co má dělat a pokud by chtěl přepnout na nějaké další video, tak bez nutnosti znát gesto k tomu určené. Na obrázku 4.1 je ukázka přehrávače MX Player, který má právě dobře udělaná dotyková gesta pro ovládání, jenže toto je nevhodné pro mé řešení a proto jsem se touto cestou nevydal.

Dále pokud navrhujeme ovládání, tak je nejdříve potřeba se zaměřit na nejdůležitější funkce a ujistit se, že při různých rozlišeních a posunech bude vše odpovídat a vše bude na svém místě a dostupné pro ovládání (a využít poskytnutých technik z dané technologie k tomu určených). Pak můžeme přidávat různé efekty a animace, ale musíme pamatovat na dodržování konzistentního vzhledu. Postupně pak také přidáváme další funkce, ale tak aby vzhled byl pořád jednotný a konzistentní a počtem funkcí a prvků jsme si neodstranili jednoduchost a přehlednost. Taktéž nesmíme zapomínat při navrhován všech efektů a animací, aby se naopak nestali záporem a něčím co uživatele bude otravovat a nebudou mít rádi (musíme si dát pozor na rychlost, plynulost, rušivost a další vlastnosti animací a efektů). Pro studium, jak navrhovat dotykové rozhraní, byla velice přínosná kniha Brave NUI world[9].

Při uvažování nad tím, co bych za funkce využil u takové to aplikace a jakým způsobem bych si obsah chtěl přehrávat, jsem se také nezapomněl zeptat i dalších uživatelů, kteří by obrazovku využívali. Výsledkem bylo několik případů užití z nichž jsem poté vycházel



Obrázek 4.1: Ovládací gesta pro MX Player pro Android

při návrhu řešení a jednotlivé požadavky jsem rozdělil do několika skupin, které proberu v následujících sekcích.

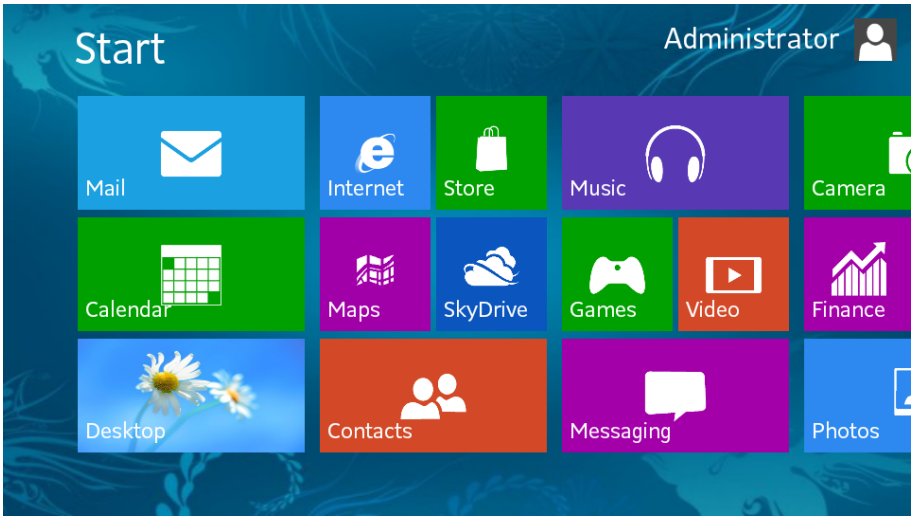
4.2 Náhled videí před začátkem přehrávání

Pokud jde o návrh zobrazení menu, tak tady bych za jednu z nejzajímavějších věcí na mé aplikaci považoval přehrávané náhledy videí. Uživatelé si velice často vybírají obsah podle jeho náhledu a to, ať už jde o prohlížení fotek v průzkumníku nebo třeba při výběru videa na webové stránce. Tyto náhledy nás mohou velice ovlivnit a často díky nim si vybereme co si prohlédneme nebo co si pustíme. Přemýšlel jsem nad klasickým náhledem, kdy bych z videí vybral určitou minutu a podle ní bych vytvořil náhled. Toto řešení by ovšem nepřineslo nic nového a proto jsem přišel s nápadem, kdy by se náhledy k jednotlivým videím přehrávaly. Už tady mi bylo jasné, že to po stránce výkonostní nebude nejjednodušší řešení, ale i přesto jsem to chtěl vyzkoušet a zjistit, jak dobře by takové řešení mohlo fungovat.

4.3 Náhledy videí pomocí WPF MediaKit

Když jsem se díval na defaultní ovládací prvek pro přehrávání videí ve WPF, což je MediaElement, tak mi bylo hned jasné, že budu muset použít nějaký jiný přístup. Třída MediaElement totiž obsahuje velké množství vlastností, metod a dalších věcí, které sice práci s ní zjednodušují a v mnoha případech pomáhají, ale v mém případě, kdy jsem jich chtěl zobrazit několik desítek najednou, tak by byly na obtíž a zbytečně ubíraly výkon aplikace. Našel jsem tedy WPF MediaKit[1], což je knihovna s ovládacími prvky postavená nad DirectShow a MediaFoundation. DirectShow je multimediální framework a API poskytované Microsoftem pro vývojáře pro usnadnění práce s multimediálním soubory a jejich streamováním. MediaFoundation poté nahrazuje DirectShow v OS Windows Vista a novějších. Tato knihovna, kromě jiných prvků, přichází právě s náhradou za MediaElement, pro kterou

[1]Knihovnu je možné si stáhnout na http://wpfmediakit.codeplex.com/



Obrázek 4.2: Ukázka Windows 8 zobrazení Metro

používá označení MediaUriElement. Poté jsem přemýšlel, jakým způsobem zobrazit několik takovýchto náhledů a v tomto ohledu mi přišlo zajímavé zkusit vytvořit seznam náhledů podobný jako seznam jednotlivých aplikací ve Windows 8 stylu zvaný Metro. A použít podobného rolování pro procházení si náhledů.

4.4 Konverze videí pomocí FFmpeg

I přesto, že chci vyměnit poměrně náročný MediaElement za méně náročný MediaUriElement, tak přehrávat zároveň několik (předpokládám až 20) náhledů videí v HD rozlišení by bylo značně náročné, což by také většina dnešních počítačových sestav nestíhala a proto jsem se rozhodl využít pro videa možnost vytvořit prvně konvertovaný náhled s menším rozlišením, který se bude přehrávat v menu a až poté co si uživatel vybere video, tak se mu začne přehrávat originální video s původním rozlišením.

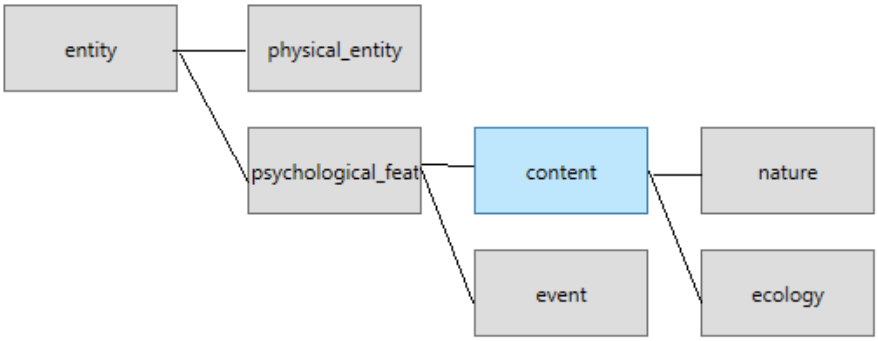
Pro konverzi je dostupná řada řešení v C# využívajících různých knihoven mezi kterými jsem si vybral FFmpeg a to z duvodů rozšířenosti, jednoduchosti práce a snadnosti implementace. FFmpeg[2] je kolekce svobodného softwaru umožňujícího nahrávání, konverzi a streamovaní digitálního zvuku a obrazu. Kolekce zahrnuje libavcodec – nejdůležitější knihovnu pro kompresi audia a videa.

4.5 Filtrování obsahu podle klíčových slov a kategorií

Jednou ze složitějších částí u obecného řešení je vymyšlení menu a usnadnění výběru videa uživateli. Předem neznáme, jaká budeme mít k dispozici videa a tedy ani jakým způsobem bychom je mohli roztřídit. V tomto ohledu nám již musí pomoci autor obsahu, který nám musí nějakým způsobem říct, co video obsahuje příp. kam patří (do jaké kategorie ho lze zařadit).

K tomu jsem vymyslel dva způsoby a oba jsem implementoval. První způsob je jednoduchý a slouží pro určení hlavní kategorie videa, kdy uživatel jednoduše v dané složce,

[2]Domovská stránka http://www.ffmpeg.org/



Obrázek 4.3: Ukázka spojení pojmů pomocí WordNetu

ze které aplikace načítá videa, dá video do podsložky, která bude určovat kategorii. Druhé řešení počítá s tím, že autor přidá k videu seznam několika klíčových slov, která se poté zobrazí uživateli a ten si pomocí nich bude moci vyfiltrovat daný obsah. Tyto dva způsoby jsem se rozhodl, že uživatel bude moci kombinovat a tím mu ještě více pomohu si vybrat video, které ho zajímá.

4.6 Hierarchický strom pomocí WordNetu

Nejdříve jsem plánoval řešení pro klíčová slova pomocí WordNetu. WordNet[11] je lexikální databáze pro anglický jazyk vyvíjená od roku 1985 týmem okolo profesora psychologie George Armitage Millera v laboratoři kognitivních věd na Princetonské univerzitě. WordNet seskupuje slova do synonymických řad zvaných synsety (anglicky synsets), poskytuje krátké obecné definice jejich významu a zachycuje různé sémantické vztahy, které mezi synsety existují. Právě těchto sémantických vztahů jsem chtěl ve svém řešení využít. Konkrétně dvou sémantických vztahů a to nadřazený a podřazený pojem. Dobrým příkladem by mohl být například slovo pes a nadřazeným pojmem pro slovo pes je psovitá šelma. Ve svém aplikaci jsem toho chtěl využít k tomu, aby tvůrce videa přidal několik konkrétních slov, která dané video vystihují a ze všech klíčových slov by se poté sestavil hierarchický strom pomocí nadřazených a podřazených pojmů a uživatel by si filtroval videa od obecnějších slov ke konkrétním.

Později při implementaci jsem ovšem zjistil, jak i s použitím WordNetu, je poměrně těžké určovat kontext daného klíčového slova a spojit více klíčových slov pod správné nadřazené slovo. WordNet obsahuje pro daný pojem několik definic, které se mohou velice lišit a jen z daného klíčového slova nepoznáme, který přesně význam autor myslel. Např. pojem Java je ve WordNetu uveden jako programovací jazyk, ostrov a druh kávy. Každá z těchto možností má různé nadřazené pojmy a sestaví se z nich poté různý strom. Druhý problém, na který jsem narazil, bylo u nadřazených na nejvyšších úrovních, kdy často byly velice obecné a ukázali se spíše matoucí než by více pomáhaly uživateli k cíli (ukázka z mé reálné implementace na obrázku 4.3). Nakonec jsem od tohoto řešení kvůli těmto dvěma problémům upustil a místo hierarchického stromu pro klíčová slova jsem se rozhodl, že implementuji slovní mrak (anglické označení jako Word Cloud nebo také Tag Cloud).

4.7 Rychlý přístup k zajímavým položkám

Mezi časté požadavky při analýze případů užití aplikace patřily požadavky na zobrazení nejnovějšího obsahu a nejzajímavějšího obsahu. Tyto dva požadavky měly své opodstatnění. Pokud by například bylo zobrazovací zařízení často ve Vašem okolí a občas byste se zastavili a prohlédli si co nového obsahuje, tak by se Vám určitě hodilo pokud by dokázalo přehrát několik nejnovějších videí. Zase naopak pokud jste na obrazovku narazili poprvé a máte čas se podívat jen na několik videí z celé kolekce pak byste určitě využili možnost přehrát si ty nejzajímavější videa. Toto byly tedy velice důležité funkce, které prostě moje řešení muselo obsahovat.

Vybrat nejnovější videa je snadné, ale musel jsem vymyslet, jak určit nejzajímavější videa. Nakonec jsem se rozhodl pro dva způsoby. Prvním způsobem určí video jako zajímavé dodavatel obsahu, který video označí jako důležité a poté je zobrazeno mezi důležitými. Druhým způsobem jsem chtěl, aby uživatelé mohli vybrat nejzajímavější videa a proto při přehrávání videa je možné dát videu hvězdičku (a tedy bod) a v nejzajímavších videí jsou ta, která mají nejvíce bodů.

Při rozhodování kam tuto funkčnosti do vzhledu zařadit jsem se rozhodl pro spodní okraj obrazovky, kdy prvně jsou vidět jenom dané označení pro jednotlivé skupiny videí a po kliknutí vyjede nabídka s videi sežazenými podle daných hodnot. Tuto nabídku jsem se rozhodl takto oddělit od zobrazovaných náhledů a jejich kategorií a klíčových slov, protože od uživatelů jasně vycházelo, že by chtěli mít možnost rychle si nechat zobrazit nejnovější či nejzajímavější videa ze všech kategorií a bez nějaké filtrace.

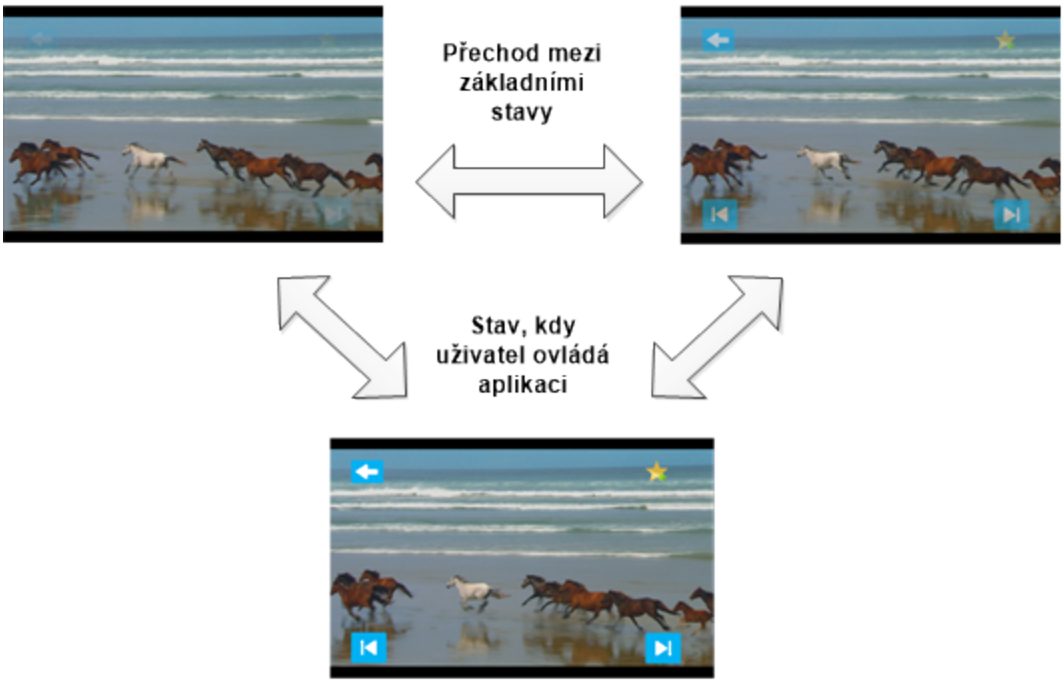
4.8 Dokázání interaktivity uživateli

Vzhledem k tomu, že jedním z požadavků bylo, aby aplikace po nějaké době bez interakce začala sama přehrávat videa, tak je potřeba také náhodným kolemjdoucím uživatelům ukázat, že mohou obrazovkou ovládat a že si mohou vybrat, co vlastně chtějí přehrávat. Jednoduchou ikonu někde v rohu jsem zavrhnul, protože neexistuje žadná, která by byla jednotnotná pro všechny platformy a která by jasně ukázala, že dané zařízení je možné ovládat dotykem. Z této nejednotnosti pak plynulo nevhodnost využití, protože uživatel by nemusel vědět co znamená nebo dokonce by si mohl myslet, že je součástí videa.

Nakonec jsem tedy vybral řešení, kdy jsou zobrazeny všechny ovládací prvky. Vzhledem k tomu, že by při normálním zobrazení překáželi při sledování obsahu (překrývali by důležité části, rušili by apod.), tak jsem vybral způsob, kdy po většinu času jsou prvky zobrazeny s velkou průhledností a po uplynutí určité doby je pomocí animace změněna průhlednost na menší a prvky jsou na malý okamžik vidět daleko více. Nakonec pokud uživatel bude ovládací prvky využívat, tak budou zobrazeny bez průhlednosti. Ukázka změny průhlednosti v mém řešení je vidět na obrázku 4.4.

4.9 Výběr ovládacích prvků pro přehrávání

Nakonec jsem se musel rozhodnout, které ovládací prvky budou potřeba a které mohu s klidem vypustit. Pro příklad jsem si vzal běžný přehrávač multimediálního obsahu a postupně jsem si řekl, jestli daný prvek bude potřeba nebo bude spíše na obtíž. Kupříkladu jsem vypustil změnu hlasitosti, která by mohla být zneužívána a někteří z uživatelů by ji mohli nastavovat na nejvyšší úroveň a pak by moje aplikace (a potažmo tedy celé zobrazovací



Obrázek 4.4: Ukázka různého využití průhlednosti u ovládacích prvků

zařízení) spíše rušila a otravovala kolemjdoucí. Dále pak podobně zbytečné by mohlo být například tlačítko pro pozastavení a znovu spuštění přehrávání. Postupně jsem se tedy výběr potřebných tlačítek dostal až k finálním 4, které tedy slouží pro následující video, začátek videa/předchozí video, zpět do menu a přidání hvězdičky k videu (započítává se do hodnocení videa).

Kapitola 5

Implementace aplikace

Jakmile jsem měl návrh aplikace hotový (viz. předchozí kapitola) vytyčil jsem si několik opěrných bodů, které aplikace musí splňovat a které postupně chci implementovat. Tyto body se s postupem vývoje lehce měnily a měnil se také postup, jakým způsobem jsem je implementoval, ale při začátku vývoje vypadaly následovně:

- Zobrazení a přehrávání náhledů videí
- Vybrání přehrávaného videa a základní ovládací prvky
- Jednoduchá kategorizace
- Využití klíčových slov pro možnost filtrace
- Načtení informace o videu z XML
- Pěkné a jednoduché grafické prostředí (vhodné pro dotykové ovládání)

Jelikož jsem byl ve WPF nováčkem, tak mi při implementaci velice pomohla kniha Mistrovství WPF[7], kterou mohu jen doporučit. Začal jsem se základní kostrou WPF aplikace (dostupnou ve VS), kterou jsem postupně doplňoval o potřebné funkce, kdy jsem prvně začínal s těmi jednoduššími. Například aplikace by měla fungovat obecně pro různá zařízení a s různými druhy video obsahu, takže by také aplikace měla mít možnost změnit některé globální nastavení a právě proto jsem implementoval načítání některých nastavení z textového souboru. Soubor se jmenuje `settings.config` a obsahuje nastavení jako cestu, ze které se bude načítat obsah nebo parametry pro proces FFmpeg s jakými se budou konvertovat videa do náhledů. Toto je velice jednoduché v každém jazyku a proto tohle byla jedna z prvních funkcí, kterou jsem implementoval a pak jsem ji využil u konverze samotného videa, kterou popisuji v následující sekci.

5.1 Konverze videí pomocí FFmpeg

Jak už jsem se zmínil v návrhu, tak videa je potřeba konvertovat a jako nástroj jsem si vybral program FFmpeg. Tento program je potřeba volat pomocí třídy Procesu[1]. Na obrázeku 5.1 jsou vidět pomocné metody, které jsem si pro tento úkol vytvořil. První metoda vytváří samotný proces pomocí jména souboru a parametrů. Zatímco druhá metoda, již spouští vytvořený proces, čte z něj informace a čeká na konec konverze a ukončení procesu.

[1]Infomrace o této třídě, dostupných parametrech a metodách na http://msdn.microsoft.com/en-us/library/system.diagnostics.process(v=vs.110).aspx

```
public class ProcessHelper
{
        public static Process CreateConvertProcessForFile(string
            fileFullName, string ffmpegParams)
        {
                var proc = new Process { StartInfo = { FileName = "ffmpeg" }
                     };
                proc.StartInfo.Arguments = "-i \"" + fileFullName + "\" " +
                    ffmpegParams + " \"" + System.IO.Path.GetDirectoryName(
                    fileFullName) + "\\" + System.IO.Path.
                    GetFileNameWithoutExtension(fileFullName) + ".sample.wmv
                    \"";
                proc.StartInfo.RedirectStandardError = true;
                proc.StartInfo.UseShellExecute = false;
                proc.StartInfo.CreateNoWindow = true;
                return proc;
        }

        public static void DoConvertingProccess(string fileFullName, string
            ffmpegParams)
        {
                var proc = CreateConvertProcessForFile(fileFullName,
                    ffmpegParams);
                if (!proc.Start())
                {
                        Console.WriteLine("Error starting");
                        return;
                }
                var reader = proc.StandardError;
                string line;
                while ((line = reader.ReadLine()) != null)
                {
                        Console.WriteLine(line);
                }
                proc.Close();
        }
}
```

Obrázek 5.1: Pomocné třídy pro vytváření procesů pro konverzi videa

Pro snažší konverzi videa jsem si ještě vytvořil pomocnou třídu BackgroundQueue, která má za úkol 2 věci. Zaprvé zaručuje, že se bude konvertovat vždy jen jedno video a ostatní procesy se budou přiřazovat do fronty a jakmile se dokončí konverze jednoho videa, tak se začne konvertovat další (pokud tedy ještě nějaké video je potřeba zkonvertovat). A za druhé se musí postarat o to, aby konverze videa proběhla v samostatném vláknu mimo hlavní. Tato třída využívá moderní vlastnosti a třídy jazyka C#. Obsahuje metodu QueueTask, která příjímá jako parametr akci, která se má provádět a vrací třídu Task, která reprezentuje

asynchronní operaci (právě ta nám bude zaručovat výpočet mimo hlavní vlákno)[4]. Abych docílil provádění jen jedné konverze, tak jsem využil klíčové slovo lock, které pracuje jako synchronizační monitor. V jazyku C# existují i konstrukce Monitor.Enter a Monitor.Exit, které mají stejný význam, ale lock zpřehledňuje kód a zároveň i je přeložen s použitím bloků try a finally na kód uvnitř konstrukce lock. Dále jsem si v třídě BackgroundQueue definoval i generickou verzi zmíněné metody a ještě pomocnou metodu pro zjištění zda je fronta prázdná.

```
public class BackgroundQueue
{
        private Task previousTask = Task.FromResult(true);
        private object key = new object();

        public Task QueueTask(Action action)
        {
                lock (key)
                {
                        previousTask = previousTask.ContinueWith(t => action
                            ()
                        , CancellationToken.None
                        , TaskContinuationOptions.None
                        , TaskScheduler.Default);
                        return previousTask;
                }
        }

        public Task<T> QueueTask<T>(Func<T> work)
        {
                lock (key)
                {
                        var task = previousTask.ContinueWith(t => work()
                                        , CancellationToken.None
                                        , TaskContinuationOptions.None
                                        , TaskScheduler.Default);
                        previousTask = task;
                        return task;
                }
        }

        public bool IsEmpty()
        {
                return previousTask.IsCompleted;
        }
}
```

Obrázek 5.2: Třída BackgroundQueue slouží pro uchování fronty požadavků a její postupné zpracování

Následné použití této třídy je velice jednoduché. Stačí volat metodu QueueTask nad globální instancí třídy BackgroundQueue a předávat ji jako argument metodu, která se má provést (tedy konverze videa). Toho docílíme využitím delegátů, které nám slouží k reprezentaci reference na metodu. Zároveň při volání metody QueueTask využijeme metodu ContinueWith, která nám stanoví, co se má dělat až se video zkonvertuje (tedy v našem případě se zavolá metoda pro přidání náhledu videa). Nakonec už jen nastavím viditelnost pro prvek značící, že probíhá konverze.

```
if (!File.Exists(videoClass.SamplePath()))
{
        bgQueue.QueueTask(() => ProcessHelper.DoConvertingProccess(item.
            FullName, Settings.SampleSettings))
                .ContinueWith((task => AddPreviewMediaForCreatedSample(item.
                    FullName)));
        busy.Visibility = Visibility.Visible;
}
```

Obrázek 5.3: Ukázka využití třídy BackgroundQueue

Jakmile je konverze dokončena a je zavolána metoda pro přidání nového náhledu, tak kromě jiného i zkontroluji, zda je prázdná fronta a pokud ano, tak mohu schovat grafický prvek, který značí, že probíhá konverze. Toto se musí provést pomocí třídy Dispatcher a zavoláním metody Invoke, protože pracuji mimo hlavní vlákno, které se stará o GUI a mimo toto vlákno není povoleno nijak měnit vzhled aplikace.

```
if (bgQueue.IsEmpty())
                Dispatcher.Invoke(new Action(() => busy.Visibility =
                    Visibility.Hidden));
```

Obrázek 5.4: Využití Dispatcheru pro změnu grafického prvku mimo hlavní vlákno

5.2 Grafické prvky

Kvůli přehlednosti, znovupoužitelnosti a lepší možnosti úprav jsem nové grafické prvky, které jsem si definoval pro svoji aplikaci přesunul do odděleného projektu ve Visual Studiu. Projekt obsahuje hlavně dva styly. První pro tlačítka, kterými se vybírají kategorie a druhý styl pro ovládací prvek TabControl, který je použit pro vysouvací nabídku s rychlím přístupem k nejnovějším videím apod.

Pokud se podíváme na část kódu použitou pro stylování TabItem položky z prvku TabControl, tak kromě jiného uvidíme také využití Konvertoru pro získání jednotlivých segmentů cesty, která je potom využitá pro efekt vlnovitého okraje u jednotlivých tlačítek.

Díky XAMLu a práci s ResourceDictionary je stylovaní a znovupoužitelnost vytvořených vzhledů velice jednoduchá a efektivní. Oba styly jsem pomocí následujících řádků kódu přiřadil jako defaultní pro dané prvky a pak už jen přidal do projektu, který obsahuje samotnou aplikaci. Oddělení grafických prvků do samostaných souborů je výhodné nejenom kvůli znovupoužitelnosti, ale také můžeme snadno upravit grafických vzhled jednotlivých prvků bez toho, abychom ovlivnili zbytek funkčnosti aplikace.

```
<Grid x:Name="grd">
        <Path x:Name="TabPath" StrokeThickness="2"
                        Margin="{Binding ElementName=TabItemContent,
                            Converter={x:Static c:ContentToMarginConverter.
                            Value}}"
                        Stroke="{StaticResource BorderBrush1}"
                        Fill="{StaticResource TabItemPathBrush}">
                        <Path.Data>
                        <PathGeometry>
                        <PathFigure IsClosed="False" StartPoint="1,0"
                            Segments="{Binding ElementName=TabItemContent,
                            Converter={x:Static c:ContentToPathConverter.Value
                            }}">
                        </PathFigure>
                        </PathGeometry>
                        </Path.Data>
                        <Path.LayoutTransform>
                                <ScaleTransform ScaleY="-1"/>
                        </Path.LayoutTransform>
        </Path>
        <Rectangle x:Name="TabItemTopBorder" Height="2" Visibility="Visible"
                        VerticalAlignment="Bottom" Fill="{StaticResource
                            BorderBrush1}"
                        Margin="{Binding ElementName=TabPath, Path=Margin}" /
                            >
        <ContentPresenter x:Name="TabItemContent" ContentSource="Header"
                        Margin="50,2,10,2" VerticalAlignment="Center"
                        TextElement.Foreground="{StaticResource
                            ForegroundBrush}"/>
</Grid>
```

Obrázek 5.5: Základ prvku TabItem v XAMLu

```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
    presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="Styles/TabItemDictionary.xaml" />
        <ResourceDictionary Source="Styles/CategoryButtonDictionary.xaml" />
    </ResourceDictionary.MergedDictionaries>
    <!-- Default Styles -->
    <Style TargetType="TabItem" BasedOn="{StaticResource TabItemStyle}" />
    <Style TargetType="Button" BasedOn="{StaticResource CategoryButtonStyle
        }" />
</ResourceDictionary>
```

Obrázek 5.6: Přiřazení vytvořených stylů jako defaultní pro dané prvky

```
<ResourceDictionary>
        <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="/VideoPlayerStyle;component/
                    VideoPlayerDesignDictionary.xaml" />
        </ResourceDictionary.MergedDictionaries>
<ResourceDictionary>
```

Obrázek 5.7: Přidání ResourceDictionary do hlavního projektu

5.3 Využití dotazovacího jazyka LINQ

Jednou z novinek v jazyce C# ve verzi 3.0 bylo přidání jazyka LINQ (anglicky Language Integrated Query)[8], který přináší nový způsob dotazování se nad daty. Při používání LINQu je možné se setkat se dvojí syntaxí. První rozšířená syntaxe je více upovídaná a ukazuji ji na příkladu 5.8. Druhá využívá rozšiřujících metod a lambda výrazů.

První metodu jsem vybral pro jednoduchou ukázku možného využití, která nám zkracuje zápis a zlepšuje čitelnost. Jedná se vlastně o funkci, kdy si seřadíme sestupně seznam tříd podle data přidání a poté si vyberem přesný počet z tříd, které potřebujeme.

```
public static IEnumerable<VideoClass> GetNewestVideos(List<VideoClass>
    videoClasses, int numberOfItems)
        {
            var list = (from v in videoClasses
                        orderby v.Added descending
                        select v).Take(numberOfItems);
            return list;
        }
```

Obrázek 5.8: Příklad LINQu pro zjištění nejnovějších videí

Druhý příklad je už poměrně složitější na první pohled, ale pokud člověk zná syntaxi LINQu a pomocných metod, tak je tato metoda hned jasná. Jedná se o metodu, která vrací všechny soubory, které mají určitou příponu ze seznamu povolených přípon a jsou v dané složce nebo její podsložce. Na prvním řádku si jenom vytvoříme HashSet[2], který nám dovolí ignorovat velikost písmen v příponě. Druhý řádek už je zajímavější, metoda EnumerateDirectories() vratí všechny podsložky dané složky a my si tyto jednotlivé složky rekurzivně procházíme a poté jejich jednotlivé výsledky vrátíme v jednom seznamu. Na třetím řádku poté projdeme všechny soubory v aktuální složce a zjistíme, které mají danou příponu a tyto poté přidáme do seznamu výsledků.

5.4 Přehrávání náhledů videí

Pro lepší výkon aplikace jsem implementoval jednoduchou funkci, která ovšem velice pomůže při zrychlení aplikace. Je samozřejmě zbytečné přehrávat všechny náhledy videí zároveň,

[2]Třída HashSet slouží k uchování seznamu objektů, kdy nám nezáleží na pořadí objektů, chceme aby obsahovala jedinečné záznamy a potřebujeme rychlí přístup k jednotlivým položkám.

```
public static List<FileInfo> GetFilesByExtensions(this DirectoryInfo
    dirInfo, params string[] extensions)
        {
            var allowedExtensions = new HashSet<string>(extensions,
                StringComparer.OrdinalIgnoreCase);
            var results = dirInfo.EnumerateDirectories().SelectMany(dir =>
                GetFilesByExtensions(dir, extensions)).ToList();
            results.AddRange(dirInfo.EnumerateFiles().Where(f =>
                allowedExtensions.Contains(f.Extension)));
            return results;
        }
```

Obrázek 5.9: Příklad LINQu pro rekurzivní prohledání složek

protože často většina z nich ani není vidět a proto pomocí následující funkce zjistím, která videa jsou vidět a tedy která přehrávat a která ne. Určujeme zobrazení podle levého horního rohu elementu a proto ještě přidávám u výsledku určité posunutí, abychom přehrávali videa, která sice mají levý roh mimo obrazovku, ale větší částí jsou vidět a naopak, abychom nepřehrávali videa, kdy je vidět jen malý kousek s levým rohem. Na tomto příkladu jde hezky vidět, jak snadno se podobné úkony dělají v technologii WPF.

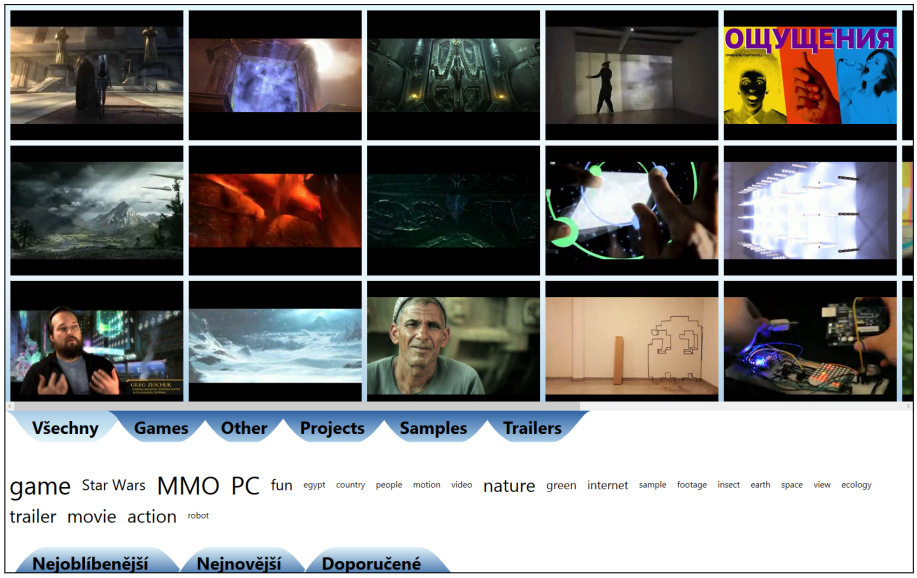
```
public static bool IsControlVisible(FrameworkElement element, ScrollViewer
    scrollViewer)
{
        var x = Canvas.GetLeft(element) - scrollViewer.HorizontalOffset;
        var y = Canvas.GetTop(element) - scrollViewer.VerticalOffset;
        return x >= -ShowOffset && x <= scrollViewer.ViewportWidth -
            ShowOffset && y <= scrollViewer.ViewportHeight;
}
```

Obrázek 5.10: Využití Dispatcheru pro změnu grafického prvku mimo hlavní vlákno

Kapitola 6

Vyhodnocení

Moje aplikace v současné době vypadá jako na obrázku 6.1. Aplikace implementuje všechny potřebné funkce pro její používání a zobrazuje náhledy videí, které jsou nahoře, řazeny do tří řádků a postupně načítány ze složky. Dále pak pomohá uživateli s výběrem videí a to pomocí řazení do kategorií (možnost vybírat si z nabídky pod náhledy), výběru klíčových slov ze slovního mraku nebo možností si zobrazit několik nejnovější či zajímavých videí, kdy se po kliknutí na dané tlačítko vysune spodní lišta s danými videi.



Obrázek 6.1: Výsledný vzhled aplikace

6.1 Testování

Aplikaci jsem měl možnost testovat na reálném zařízení, na kterém je plánováno i reálné využití této aplikace a nejprve jsem ji otestoval já, aby neobsahovala žádné chyby a mohla se pohodlně používat tak, jak jsem zamýšlel. Poté jsem si pozval několik uživatelů, kteří by patřili mezi potenciální uživatele dané aplikace a bez nějaké instruktáže jsem je postupně poprosil, aby si aplikaci vyzkoušeli a zjistili co všechno umí, jak se jim s ní pracuje a poté mi sdělili své názory.

Musím říct, že aplikace je na první pohled zaujala a díky své jednoduchosti se uživatelé hned začali orientovat v menu a bez problémů pochopili, jakým způsobem se aplikace ovládá a co je potřeba udělat proto, aby si vybrané video spustili nebo jakým způsobem se dají videa třídit či filtrovat. Při přehrávání videa uživatele velice zaujalo, že obsahuje jenom pár ovládacích prvků a také je zajímalo proč se po nějakém čase prvky více zviditelní.

Výsledné ohlasy byly velice kladné a potěšilo je, že by měli možnost například na chodbě v CVT se na zajímavá videa nejenom dívat, ale také videa si vybírat podle toho, která je zaujmou.

Jako jednu z mála výtek, kterou jsem od uživatelů dostal, byla nemožnost posouvat oblast s videi pomocí tahu prstu v oblasti s náhledy, ale nutnost prstem být přímo nad posuvníkem (anglicky scroll bar) a jedině s ním hýbat. Pokud uživatel zkoušel posouvat videa v oblasti s náhledy, tak se mu místo toho začalo video přehrávat. Tohoto problému jsem si všimnul až při testování a je to zapříčiněno tím, že se nejdříve vyvolá událost kliknutí myši na náhled před událostí posunování lišty s náhledy. Tuto vadu na kráse bych rád opravil v další verzi.

Bohužel jsem již nestihl aplikaci nasadit do reálného provozu a proto mám uživatelské podněty jenom z krátkého testování při dokončování aplikace. Toto bych ještě rád změnil a pokusil se svou aplikaci nasadit do reálného prostředí a zjišťovat, jak se celkově uživatelům líbí a ještě dále aplikaci ladit podle potřeb a využití. Myslím si, že by tato aplikace našla reálné využití na různých místech jako například výstavách, chodbách, obchodech apod., kde by bylo potřeba na obrazovce prezentovat videa s danými produkty či z dané oblasti, kde se zařízení nachází. Díky WPF a hlavně XAMLu je aplikaci možno jednodušše skinovat, takže by se dokázala snadno graficky doladit podle potřeby.

Kapitola 7

Závěr

Navrhl jsem a vytvořil aplikaci pro prezentování videí na obrazovce s dotykovým vstupem pomocí jazyku C# a technologie WPF. Jsem velice hrdý na to, že se mi podařilo implementovat všechny požadavky, které jsem si stanovil a které byly potřeba, aby aplikace mohla reálně fungovat. Kvůli zaměření mé aplikace na přehrávání multimediálního obsahu jsem navrhnul několik způsobů ovládání, které uživateli pomáhají s výběrem právě toho videa, které hledá.

Nejvíce mě mrzí, že jsem již aplikaci nestihl nasadit do reálného prostředí a kvůli tomu jsem pak nemohl posbírat údaje o interakci od opravdu náhodných uživatelů, které zobrazovací zařízení zaujme a místo toho jsem si pozval jen několik uživatelů na otestování reálného zařízení. Z toho co jsem mohl zjistil od uživatelů, tak je aplikace zaujala a rádi by ji viděl a vyzkoušeli v reálném prostředí na chodbě v CVT. V tomto bodě se mi tedy bohužel nepodařilo splnit zadání své bakalářské práce.

Pokud jde o další rozšiřování aplikace, tak už v tuto chvíli mám sepsaných několik poznámek, které bych rád do své aplikace ještě implementoval. Mezi ty nejzajímavější patří například možnost přidání prezentace k danému videu, kterou by si uživatel mohl navíc prohlédnout, pokud by ho video zaujalo.

Bylo zajímavé pracovat s jazykem C# a technologií WPF a rád bych pokračoval dále v úpravě této své aplikace s využitím daných technologií. Věřím, že se mi podaří aplikaci nasadit do reálné prostředí a vidět ji používanou uživateli.

Literatura

[1] Abrams, B.: What is managed code? [online]. Dostupné z http://blogs.msdn.com/b/brada/archive/2004/01/09/48925.aspx, 2004-01-04 [cit. 2013-04-05].

[2] Android: Android Design [online]. Dostupné z http://developer.android.com/design/index.html, 2013-02-01 [cit. 2013-04-15].

[3] Apple: iOS Developer Library [online]. Dostupné z https://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html, Poslední aktualizace 2013-05-01 [cit. 2013-04-15].

[4] MSDN: Task Parallelism (Task Parallel Library) [online]. Dostupné z http://msdn.microsoft.com/en-us/library/dd537609.aspx, 2013-02-01 [cit. 2013-04-28].

[5] MSDN: Overview of the .NET Framework [online]. Dostupné z http://msdn.microsoft.com/en-us/library/zw4w595w.aspx, 2013-04-01 [cit. 2013-04-05].

[6] MSDN: Data Binding Overview [online]. Dostupné z http://msdn.microsoft.com/en-us/library/ms752347.aspx, 2013-04-07 [cit. 2013-04-12].

[7] Petzold, C.: *Mistrovství ve Windows Presentation Foundation*. Computer Press, 2008, iSBN 978-80-251-2141-2.

[8] Pialorsi, P.; Russo, M.: *Microsoft LINQ: kompletní průvodce programátora*. Computer Press, 2009, iSBN 978-80-251-2735-3.

[9] Wigdor, D.; Wixon, D.: *Brave NUI world: designing natural user interfaces for touch and gesture*. Morgan Kaufmann, 2011, iSBN 978-0-12-382231-4.

[10] Wikipedia: Windows Presentation Foundation - Architecture [online]. Dostupné z http://en.wikipedia.org/wiki/Windows_Presentation_Foundation#Architecture, Poslední aktualizace 2013-03-12 [cit. 2013-04-01].

[11] Wikipedia: WordNet [online]. Dostupné z http://cs.wikipedia.org/wiki/WordNet, Poslední aktualizace 2013-04-05 [cit. 2013-05-01].

Příloha A

Obsah DVD

Přiložené DVD obsahuje tyto části:

- Aplikace - obsahuje projekt pro Visual Studio 2012 včetně zdrojových kódů aplikace a potřebných knihoven pro překlad
- Text - obsahuje PDF soubor bakalářské práce (a zdrojové soubory v latexu)
- Plakat - obsahuje PDF soubor s plakátem k aplikaci
- Video - obsahuje video soubor s prezentací vlastní aplikace