Multimédia v technologii Silverlight

Bakalářská práce

Miroslav Veith

Vedoucí práce: Mgr. Miloš Prokýšek

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

Rok 2010

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval/-a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne 31. 3. 2010

Zadání práce

Student zmapuje možnosti a omezení práce s multimédii v technologii WPF a Silverlight. Popíše společné rysy Silverlight a WPF. Student se v práci zaměří především na problematiku streamingu a provázání s technologií DirectX. Dále se student zaměří na práci se zdroji videa v prostředí WPF a Silverlight. Student vytvoří ukázkové miniaplikace pro ověření funkčnosti popisovaných postupů.

Anotace

Tato práce se zabývá analýzou možností technologie Silverlight při přehrávání streamovaného obsahu. Součástí práce je vytvoření ukázkové aplikace. Tato aplikace byla využita při ověřování vlivu přehrávání streamovaného obsahu na zatížení systému

Dále se tato práce zabývá možnostmi technologie Silverlight v oblasti práce se zařízeními na straně klienta. Mezi tato zařízení patří webkamery, mikrofony atd. Cílem práce je také testování možností Silverlightu s cílem vytvořit aplikaci, umožňující uživatelům video hovory, která by těchto zařízení využívala.

Abstract

This work deals with analysis of possibilities of technology Silverlight on playing streaming content. Part of the work is creating of sample application. This application will be used for examination of influence of playing streaming content on burden of system.

Next this work deals with possibilities of technology Silverlight in domain of working with devices on client side. Among these devices belong Webcams, Microphone etc. A goal of this work is also an attempt to create Silverlight application enabling to users video talks which would be using of these possibilities.

Obsah

1 Úvod

Silverlight je poměrně nová technologie, která vznikla v roce 2007 a je vážným konkurentem do té doby bezkonkurenčního Flashe, ten se stal samozřejmou součástí webových stránek. Na to zareagoval Microsoft vznikem technologie založené na WPF, Silverlightem. Silverlight je odlehčenou a mírně upravenou verzí WPF, určenou pro využití na webových stránkách. Ne všechny možnosti WPF se dají využít i v Silverlightu a to především z bezpečnostních důvodů. Dalšími jsou velikost pluginu, použitelnost i na počítačích, kde není nainstalovaný .NET Framework atd.

Technologie Silverlight je zajímavá v mnoha ohledech. Nejméně dokumentovanou a popsanou oblastí jsou možnosti přehrávání streamovaného videa, přestože je to oblast velice zajímavá a také důležitá z hlediska webových aplikací.

Streaming videa je v současnosti velice rychle se rozšiřující oblastí webových technologií. Různé televize na svých stránkách poskytují archivy vlastních pořadů, další poskytují své vysílání online, sportovní události jsou přenášeny živě na stránkách věnovaných sportu atd. Navíc lidé ve velké míře využívají služby pro sdílení videa jako je youtube.com a nebo český stream.cz. Obrovská a rostoucí popularita těchto služeb jasně napovídá, že streaming videa by rozhodně neměl být opomíjen, spíš naopak. Služby, které jsou zmíněny výše (Youtube.com, Stream.cz…) využívají pro přehrávání streamovaného videa technologii Flash, jen málokde (tv.hokej.cz) se zatím využívá Silverlight. A to i přesto, že Silverlight v této oblasti nabízí skvělé možnosti.

V první části si proto práce klade za cíl analyzovat možnosti přehrávání streamovaného obsahu v Silverlightu za použití různých kompresních formátů a transportních protokolů a porovnat přehrávání tohoto obsahu z hlediska zatížení systému.

Dílčím cílem v této části práce je rovněž empirické ověření vlivu podpory GPU v Silverlightu na celkové zatížení systému při přehrávání streamovaného obsahu.

Dále se práce zabývá různými metodami poskytování streamovaného videa klientovi. Jsou to klasický streaming, Progressive Download a speciální druhy poskytování streamovaného obsahu – například Smooth Streaming.

Dalším dílčím úkolem v této první části práce bude prozkoumat možnosti týkající se ochrany obsahu a vlastnických práv autorů a majitelů videa, jedná se o tzv. Silverlight DRM – Digital Rights Management.

Ve druhé části si tato práce klade jako hlavní cíl zjistit, zda Silverlight dokáže pracovat s webkamerami a dalšími zařízeními na straně klienta, ať už se jedná o integrované webkamery, kamery připojované k počítači klienta nebo mikrofony. Pokud to v aktuální verzi možné není, tak se práce pokusí odpovědět na otázku: Jak to s podporou zařízení na straně klienta vypadá směrem do budoucnosti? Pokud z předchozího zkoumání vyplyne, že to je možné už za současného stavu, bude vytvořena ukázková Silverlight aplikace. Tato aplikace bude využívat všechny dosud zjištěné poznatky a bude umožňovat komunikaci přes internet. Ukázková aplikace bude pracovat s videem snímaným webkamerou a přenášeným mezi komunikujícími, to znamená, že bude umožňovat video hovory mezi klienty.

2 Co je Silverlight

Tvůrci technologie Silverlight představují svůj produkt takto (3):

„Microsoft® Silverlight™ je nejmodernější technologie pro internetové prohlížeče. Je to platforma určená pro tvorbu dynamického online obsahu a interaktivní práce s ním. Kombinuje text, vektorovou i bitmapovou grafiku, animace a video.

Pomocí malé stažitelné komponenty (plug-in) umožní interaktivní ovládání her nebo aplikací a přehrávání multimédií ve většině současných webových prohlížečů (Internet Explorer, Firefox, Safari, Opera, Chrome) na platformách Windows a Mac OS X. Na Linuxu je dostupný pod názvem Moonlight, vyvinutý společností Novell.“

Silverlight byl původně označován kódovým jménem WPF/E (2) (Windows Presentation Foundation Everywhere), což také označuje původní myšlenku této technologie. Myšlenkou, která vedla ke vzniku Silverlightu je, aby bohaté desktopové aplikace vytvořené pomocí technologie Windows Presentation Foundation, byly dostupné nejen v operačním systému Windows, ale i v dalších operačních systémech (2). Z této myšlenky vychází kódové označení WPF Everywhere, přeloženo do češtiny WPF kdekoliv.

Tato nezávislost na operačním systému Windows byla podmíněna fungováním i tam, kde není nainstalovaný .NET Framework, který je dostupný pouze ve Windows. Toho je dosaženo tím, že Silverlight nevyužívá .NET Framework, ale plug-in pro internetový prohlížeč.

Jak již vyplývá z výše citovaného, tak Silverlight je technologie pro tvorbu RIA aplikací (Rich Internet Application), tedy bohatých internetových aplikací. Tyto aplikace se snaží zmenšovat rozdíly mezi internetovými aplikacemi a desktopovými aplikacemi.

Silverlight aplikace se tvoří podobným způsobem, jako WPF aplikace, ovšem s jistými omezeními. Tato omezení vyplývají z toho, že Silverlight aplikace musí běžet v sandboxu prohlížeče (výjimkou je možnost nainstalovat aplikaci do počítače

uživatele, kdy se jedná o tzv. Out of Browser mód (3)). Tento sandbox zajišťuje, aby aplikace běžící v internetovém prohlížeči neohrožovaly zabezpečení operačního systému. Omezení Silverlightu oproti WPF plynou většinou z jiného přístupu k bezpečnosti u obou technologií. Pokud vývojář přechází na Silverlight z WPF musí s těmito omezeními počítat. Silverlight ale umožňuje vytváření aplikací pomocí XAML kódu a C# kódu (nebo pomocí jiných programovacích jazyků z platformy .NET), což vývojářům velice usnadňuje přechod z desktopových technologií (WPF, Windows Forms) nebo z ASP.NET na Silverlight.

Zkompilované Silverlight aplikace jsou zabalené v jediném souboru, který stačí vložit do webové stránky. Tyto soubory mají příponu XAP. Silverlight aplikace se dají vkládat do statického webu nebo do webových formulářů ASP.NET. Pro práci se Silverlight aplikací ve webové stránce se využívají funkce JavaScriptu, který například načte a zobrazí aplikaci, pracuje s výjimkami atd.

3 Streaming

Na otázku: „Co je to streaming?“, nejlépe odpoví popis různých metod doručování obsahu.

3.1 Metody doručování obsahu

3.1.1 Progressive Download

Progressive download nebo také postupné stahování je způsob přenosu digitálního média ze serveru ke klientovi. Při tomto způsobu přenosu se využívá většinou HTTP protokol. Při progressive downloadu klient nemusí čekat, až se stáhne celý soubor, ale může začít přehrávat část, která už byla stažena. To znamená výhodu proti stahování celého souboru, kdy klient musí počkat, než se stáhne celý soubor.

Přehrávač, který přehrává tímto způsobem, postupně stahuje data do lokální vyrovnávací paměti. Přehrávač začne soubor přehrávat v okamžiku, kdy je v lokální vyrovnávací paměti uložen dostatečný objem dat.

Výhodou tohoto přístupu je rychlost začátku stahování. V průběhu přehrávání pokračuje stahování souboru a k přehrávání jsou dostupné další části souboru. Další výhodou je, že tato metoda se dá využít i pokud má uživatel pomalé připojení k internetu, jelikož se přehrává soubor uložený na lokálním disku a ne soubor, který je přenášen z internetu. Výhodou může být také cena, neboť není potřeba žádný speciální streamovací server a stačí obyčejný webový server jako je IIS nebo Apache. Díky tomu je dosaženo větší jednoduchosti a dostupnosti.

Ovšem nevýhodou je stahování souboru na disk (přestože z jiného pohledu je to nespornou výhodou), kdy uživatel může přehrávaný soubor získat nelegálně, pokud si ho z dočasného úložiště zkopíruje. Takže tam, kde se vyžaduje nepřístupnost souborů, není vhodné používat tuto metodu. Tuto metodu také není možné použít při živém streamování (přímé přenosy atd.), protože není co stahovat a ukládat na disk. Další nevýhodou oproti klasickému streamování je určitá neefektivita, kdy je stahována i ta část souboru, kterou si uživatel nikdy nepřehraje a není plně využívána šířka pásma, přenášejí se i nepotřebná data. Nevýhodou je také to, že se stahuje jeden soubor a není možné přizpůsobit se šířce pásma uživatele.

3.1.2 Klasický streaming

Klasický streaming nebo také streamování je způsob přenosu digitálního média ze serveru ke klientovi. Při této metodě je přenášený obsah v komprimované formě přenášen ze serveru ke klientovi, kde je přehrávačem přehráván v reálném čase. Využívají se různé protokoly, které závisí na vybrané technologii a konkrétním způsobu. Uživatel nemusí čekat, až se přenášený soubor stáhne, ale může začít přehrávat ihned, když k němu dorazí datový proud souboru.

Uživatel potřebuje přehrávač, který tento datový proud dokáže dekomprimovat a přehrát. Tento přehrávač neshromažďuje doručená data na disku, ale okamžitě přehrává to, co mu bylo doručeno. Ve vyrovnávací paměti přehrávače, takzvaném bufferu, je v jednom okamžiku uložena pouze část souboru, například 5 vteřin dlouhý záznam. Velikost tohoto bufferu je ovlivněna nastavením přehrávače.

Výhodou je, že se oproti postupnému stahování, mnohem efektivněji využívá šířka pásma sítě. Toho je dosaženo tím, že se přenáší pouze ta data, jež jsou skutečně přehrávána, a lze předpokládat, že uživatel tuto část sleduje. Klasický streaming poskytuje další velkou výhodu v tom, že se dá přenášet živě, což znamená, že tento způsob je vhodný pro streamování živých sportovních přenosů a také živé streamování televizního a rádiového vysílání. U této metody je také možné využít technologii multicast, kdy jsou stejná data odesílána jen jednou, ale jsou přehrávána více uživateli. Oproti postupnému stahování se tato metoda dá využít také tam, kde je vyžadována nepřístupnost přenášených souborů. Tohoto zabezpečení obsahu se dosahuje proto, že na straně uživatele jsou uložená pouze aktuální data z bufferu a ne celý soubor.

Nevýhodou je, že uživatel potřebuje určitou minimální šířku pásma, aby k němu mohl být přenášen souvislý proud dat, protože data se nikde nehromadí. Velkou nevýhodou pro poskytovatele obsahu je cenová náročnost, protože tato metoda vyžaduje speciální streamovací server.

3.1.3 Adaptive streaming

Adaptive nebo také multi-bitrate streaming (9). Nejedná se ani tak o třetí metodu pro přenos souboru ze serveru ke klientovi, ale jedná se o zcela jiný přístup k tomuto přenosu a nelze ho zařadit ani pod postupné stahování, ani pod klasický streaming.

Hlavní rozdíl oproti předchozím metodám je ten, že adaptive streaming, jak už název napovídá, je přizpůsobivý. Přizpůsobuje se podmínkám připojení klienta tak, aby uživatel měl co nejlepší komfort při sledování videa nebo poslechu hudby. Hlavní myšlenkou celého tohoto přístupu je, aby přehrávání začalo okamžitě, byť v nižší kvalitě. Pro uživatele je lepší, když přehrávání začne v nižší kvalitě okamžitě, než když nevidí nic a musí čekat na načtení dalších dat. Tento způsob analyzuje připojení klienta a také změny stavu tohoto připojení a přenášená data upravuje v závislosti na podmínkách sítě tak, aby přehrávání na straně uživatele probíhalo nepřetržitě.

To, že přehrávání probíhá nepřetržitě je zajištěno tím, že se v závislosti na podmínkách připojení uživatele mění kvalita přenášeného videa nebo audia. Takže pokud se šířka pásma prudce sníží, musí se snížit kvalita přenášeného obsahu tak, aby bylo možné tento obsah bez přerušení přenášet ze serveru ke klientovi. Proto je vyžadováno, aby na serveru nebylo video kódováno pouze v jedné kvalitě, ale musí být ve více kvalitách s různými datovými toky (v anglické literatuře se setkáme s označením bitrates (7, 8, 9)). Odtud pochází označení multi-bitrate streaming (9). Hodnota datových toků musí být takové, aby se přenášení mohlo přizpůsobit největšímu počtu možných scénářů. Při velice dobrém připojení a široké šířce pásma je možné přenášet i video s velkým datovým tokem, například HD video. A zároveň, i pokud je připojení špatné, adaptive streaming umožní klientovi sledovat video bez přerušení. Přizpůsobivost adaptive streamingu je omezena na počet souborů s různým datovým tokem.

Jak už bylo řečeno, adaptive streaming nemůžeme zařadit pod žádnou z předchozích metod, protože na první pohled se jedná o klasický streaming, kdy ke klientovi proudí stálý tok dat, ve kterém je přenášena pouze sledovaná část a nic se nestahuje na disk uživatele. Ovšem u většiny implementací adaptive streamingu (9,10) se ve skutečnosti jedná o to, že přenášený soubor je rozdělený na

malé části, které jsou ke klientovi přenášeny postupným stahováním těchto malých částí. A na druhou stranu u jiných implementací (10) se využívá klasické streamování, kdy se server v závislosti na stavu sítě rozhoduje, který soubor (datový tok) bude streamovat. Takže u adaptive streamingu nalezneme znaky obou výše popsaným metod a není možné jednoznačně rozhodnout, zda patří pod klasický streaming a nebo postupné stahování.

Stejně tak není jednoznačně možné popsat tento přístup po technické stránce, protože různé implementace adaptive streamingu fungují jinak, ale jsou založeny na stejné myšlence a můžeme je zařadit do jedné skupiny – adaptive streaming. Těchto implementací v současnosti nalezneme mnoho, v dalších kapitolách bude popsána implementace od Microsoftu, která se nazývá IIS Smooth Streaming. Existují implementace i od jiných společností jako například: Flash Dynamic Streaming od společnosti Adobe, HTTP Adaptive Bitrate Streaming od společnosti Apple a Move Adaptive Stream od Move Networks. Tyto implementace nejsou navzájem kompatibilní a opět se dostáváme k tomu, že adaptive streaming není standardizovaným řešením, ale pouze pohledem na přenos dat.

Ze čtyř jmenovaných implementací všechny kromě Flash Dynamic Streaming (protokol RTMP) jsou založeny na protokolu HTTP, kdy jsou soubory rozděleny na malé části, většinou o délce 2 až 4 vteřiny (9), které je snadné stáhnout a přehrávat postupně. Soubory s různými datovými toky jsou rozděleny na časově stejně dlouhé části a přehrávač na straně klienta podle stavu připojení vybírá, ze kterého souboru stáhne aktuální část. Dělení na tyto části musí být pečlivě provedeno kódovacím programem tak, aby postupné spojování různých těchto částí bylo pro uživatele zcela neznatelné a nerušilo ho.

Nejdůležitější roli v tomto přístupu hraje v první řadě poskytovatel videa, který musí video do různých datových toků kódovat. K tomu poskytovatel musí využít program, který soubory kóduje ve správném formátu. Dále je důležitý přehrávač na straně klienta, který musí být schopen analyzovat stav sítě a zároveň přehrávat takto kódované soubory. Na druhou stranu například u IIS Smooth Streamingu nejsou kladeny žádné zvláštní nároky na server a stačí využít obyčejný webový server IIS s rozšířením, které je k dispozici zdarma. Podobně je tomu i u technologie HTTP

Adaptive Bitrate Streaming od společnosti Apple, kde je možné využít jakýkoliv webový server (nejen IIS), opět jen s doinstalováním rozšíření dostupného zdarma.

4 Silverlight a přehrávání streamovaného obsahu

4.1 Silverlight a přehrávání streamovaného obsahu

Silverlight jako technologie určená pro zobrazování bohatého obsahu – RIA (Rich Internet Application), samozřejmě umožňuje práci s videem a zvukem. Silverlight v tomto ohledu nabízí široké možnosti a mnoho velice zajímavých funkcí. Tyto možnosti a funkce jsou popsány v této kapitole.

Pro široké uplatnění technologie v oblasti přehrávání streamovaného videa a zvuku je nutné, aby technologie podporovala co nejvíce možných formátů a byla co nejvýhodnější nejen pro uživatele, kteří si video nebo zvuk přehrávají, ale také pro poskytovatele obsahu.

V Silverlightu se k přehrávání videa a audia používá prvek (třída) MediaElement. MediaElement je základním prvkem, který přehrává všechny podporované formáty, takže přehrává zvuk i video. Zároveň také podporuje všechny metody pro doručování obsahu, které jsou dostupné v Silverlightu. Pokud se hovoří o možnostech Silverlightu, jsou zároveň popisovány možnosti třídy MediaElement.

4.1.1 Podporované formáty

Silverlight podporuje širokou paletu formátů videa i zvuku. Pojmem formát je myšlen kodek, ale také kontejner. Kodek je zařízení nebo software, které poskytuje video nebo audio kompresi a dekompresi (14). Kodeky rozlišujeme na audio kodeky a video kodeky. Kontejner je obálkou audio nebo video souboru, která uchovává nejen data, ale také informace o tom, jak jsou data v kontejneru uložena (14). Podporované kontejnery a kodeky jsou vypsány v následujícím přehledu (14).

Podporované kontejnery:

- Windows Media
- MP4
- MP3
- ASX
- fMP4 – IIS Smooth Streaming
- Server-side Playlist (SSPL) – konfigurační soubory mají příponu wsx
- Vlastní kontejner – implementace abstraktní třídy MediaStreamSource, vývojáři poskytuje nezávislost na podporovaných kontejnerech a kodecích

Podporované video kodeky:

- RAW video – nejedná se o kodek, ale o nekomprimovaná video data
- YV12 - YCrCb(4:2:0) – nejedná se o kodek, ale o formát nekomprimovaných video dat
- RGBA – nejedná se o kodek, ale o formát nekomprimovaných dat, k základním barvám se přidává možnost pracovat s kanálem průhlednosti
- WMV1: Windows Media Video 7
- WMV2: Windows Media Video 8
- WMV3: Windows Media Video 9:
    - podporován pouze progresivní obsah – ne prokládaný
    - podporovány profily Simple a Main
- WMVA - Windows Media Video Advanced Profile, non-VC-1
- WVC1: Windows Media Video Advanced Profile, VC-1:
    - podporován profil Advanced
    - podporován pouze progresivní obsah – ne prokládaný
- H264 (ITU-T H.264 / ISO MPEG-4 AVC):
    - podpora kodeků H.264 a MP43
    - podporovány profily Base, Main a High
    - podporován pouze progresivní obsah – ne prokládaný

Podporované audio kodeky:

- PCM (8 nebo 16 bitová Pulse Code Modulation) – nejedná se o kodek, ale o nekomprimované audio – jednoduše se tento formát nazývá WAV
- WMA Standard:
  - Microsoft Windows Media Audio v7
  - Microsoft Windows Media Audio v8
  - Microsoft Windows Media Audio v9.x Standard
- WMA Professional:
  - Microsoft Windows Media Audio v9.x
  - Professional Microsoft Windows Media Audio v10 Professional
- MP3 – ISO MPEG-1 Layer III
- AAC – ISO Advanced Audio Coding
  - není podporován vícekanálový (5.1) zvuk

4.1.2 Podporované metody doručování obsahu

Další věcí, která je velice důležitá, jsou metody doručování audio/video obsahu ze serveru ke klientovi. Mezi 3 základní metody doručování obsahu ze serveru patří výše popsané – progressive download, klasický streaming a adaptive streaming. Silverlight podporuje všechny tyto 3 metody. Pro vývojáře ani pro uživatele využívajícího klientskou aplikaci v Silverlightu se nic nemění, ať už se použije progressive download nebo klasický streaming.

Pro progressive download stačí MediaElementu nastavit jako zdroj URL cestu k souboru, který se má přehrát. MediaElement poté z URL zjistí, že se jedná o protokol HTTP a pokusí se přehrát soubor s využitím progressive downloadu. Pokud by tato metoda selhala, tak se pokusí zjistit, zda se nejedná o klasický streaming a pokud ani to se nepodaří, nic se nepřehraje. MediaElement o takové situaci informuje pomocí události MediaFailed, která je volána také tehdy, když soubor, který se měl přehrát je v nepodporovaném formátu. Na straně serveru také není potřeba žádné speciální řešení, možné je využít jakýkoliv webový server.

Při použití klasického streamování (živého nebo na vyžádání) stačí, stejně jako u předchozí metody, nastavit jako zdroj URL ke streamovanému obsahu.

MediaElement z URL adresy zjistí, že se jedná například o protokol MMS nebo RTSP a pokusí se soubor přehrát s využitím klasického streamování, pokud to nejde, tak vyzkouší, zda se nedá použít progressive download a pokud ani to není možné, nic se nepřehraje. Na straně serveru je třeba využít Windows Media Services a Windows Server. Takže oproti předchozí metodě je tady omezení na straně poskytovatele obsahu. Na straně klienta, tedy Silverlightu, není třeba žádné rozšíření.

Používání třetí metody, adaptive streamingu, je trošku odlišné. Silverlight v současné verzi podporuje přehrávání pomocí IIS Smooth Streaming, implementace adaptive streamingu od společnosti Microsoft. Tato implementace byla vytvořena speciálně pro použití v Silverlightu. O IIS Smooth Streamingu podrobněji dále.

4.1.3 Další funkce Silverlightu při přehrávání streamovaného obsahu

Jak už bylo řečeno Silverlight nabízí různé další možnosti při přehrávání streamovaného obsahu. Tyto možnosti se týkají například zabezpečení streamovaného obsahu proti neoprávněnému použití, vytváření a používání vlastních formátů videa nebo zvuku, použití Silverlight aplikací v různých zvláštních scénářích atd. Následuje výčet těchto možností se základním popisem – podrobnější popis některých funkcí bude následovat dále:

- HD video – Silverlight uživatelům nabízí možnost sledovat video v HD kvalitě (720p a 1080p) a to dokonce v režimu celé obrazovky, využívá se podpory kodeků H.264 a VC-1
- Podpora GPU – Silverlight slibuje podporu GPU nejen pro akcelerované video, ale pro všechny bitmapové elementy včetně videa, to umožňuje mnohem plynuleji přehrávat HD video i v režimu celé obrazovky
- Out of Browser – Silverlight od verze 3 nabízí podporu pro spouštění aplikací i mimo prohlížeč
- Podpora různých operačních systémů a různých prohlížečů – Silverlight v současné době nabízí nativní podporu operačních systémů Windows a OS X, také nabízí podporu operačního systému Linux (Linuxová implementace Silverlightu - Moonlight), současně nabízí oficiální podporu prohlížečů Internet Explorer, Mozilla Firefox, Google Chrome a Safari
- DRM (Digital Rights Management) – technologie na ochranu obsahu
- Záložky – Silverlight nabízí výborné možnosti pro práci se záložkami (kapitolami) ve videu, samozřejmě video musí být kódováno nástrojem, který se záložkami umí pracovat
- Server-side Playlist (SSPL) – umožňují poskytovateli videa mít kontrolu nad přehrávaným obsahem, Silverlight ovšem nepodporuje všechny možnosti SSPL (12)
- Multicast – Silverlight společně s Windows Media Services a volně stažitelným pluginem od společnosti Qumu

HD video

Podpora HD videa slibovaná vývojáři technologie Silverlight bude v další části této práce empiricky ověřena.

Podpora GPU

Rovněž slibovaná podpora GPU bude empiricky ověřena experimentem a bude změřen její vliv na celkové zatížení systému.

Out of Browser

Jak už bylo řečeno Silverlight umožňuje spouštět aplikace i mimo webový prohlížeč. Funguje to jednoduše tak, že se aplikace běžící v prohlížeči nainstaluje na lokální počítač a může být spuštěna bez spuštění prohlížeče, ale také bez připojení k internetu. Toho se využívá při různých speciálních scénářích, kdy se video soubor stáhne do počítače uživatele a ten si ho potom může přehrát i tam, kde není dostupné připojení k internetu. Po připojení k internetu se aplikace aktualizuje a videa se například mohou také aktualizovat nebo obměnit atd.

Podpora různých OS a různých prohlížečů

Silverlight podporuje, jak už bylo zmíněno výše, různé operační systémy, v současné době to jsou (11):

- Windows 2000 (od SP4)
- Windows XP (od SP2)
- Windows Vista
- Windows 7
- Windows Server 2003
- Windows Server 2008
- Windows Server 2008 R2
- Mac OS X Mac OS 10.4.8+ (PowerPC)
- Mac OS 10.4.8+ (Intel-based)
- Linux, FreeBSD, SolarisOS – Moonlight – implementace Silverlightu od společnosti Novell

Dále Silverlight, jak bylo také zmíněno výše, oficiálně podporuje různé prohlížeče a další nejsou oficiálně podporovány, ale většinu Silverlight aplikací lze bez problémů spustit (například Opera). Podpora prohlížečů podporovaných pod operačním systémem Windows (IE8, Firefox 3, Chrome) bude také dále empiricky ověřena. Seznam oficiálně podporovaných prohlížečů (některé verze jen pod určitými operačními systémy) (11):

- Internet Explorer 6
- Internet Explorer 7
- Internet Explorer 8
- Firefox 2
- Firefox 3
- Safari 3 – pouze pro Mac OS X
- Safari 4 – pouze pro Mac OS X
- Google Chrome

DRM

Protože s rostoucí popularitou služeb poskytujících streamovaný video obsah roste také počet společností, které se na takovýchto službách rozhodly vydělávat. Tak vznikly ze strany poskytovatelů obsahu požadavky na ochranu jejich obsahu, proti neoprávněnému používání – ať už se jedná o nelegální sledování nebo o neoprávněné šíření tohoto obsahu.

Silverlight proto vychází vstříc i těmto požadavkům a nabízí několik možností, jak obsah chránit. Hlavní technologií určenou pro ochranu obsahu je DRM (Digital Rights Management), která sice vznikla už před vznikem Silverlightu, ale je se Silverlightem kompatibilní.

V Silverlightu je možné využít dva typy DRM:

- Silverlight DRM – spolupráce se serverem PlayReady a příslušnými licencemi, používá se Advanced Encryption Standard (AES)
- Windows Media DRM 10 – WMDRM 10

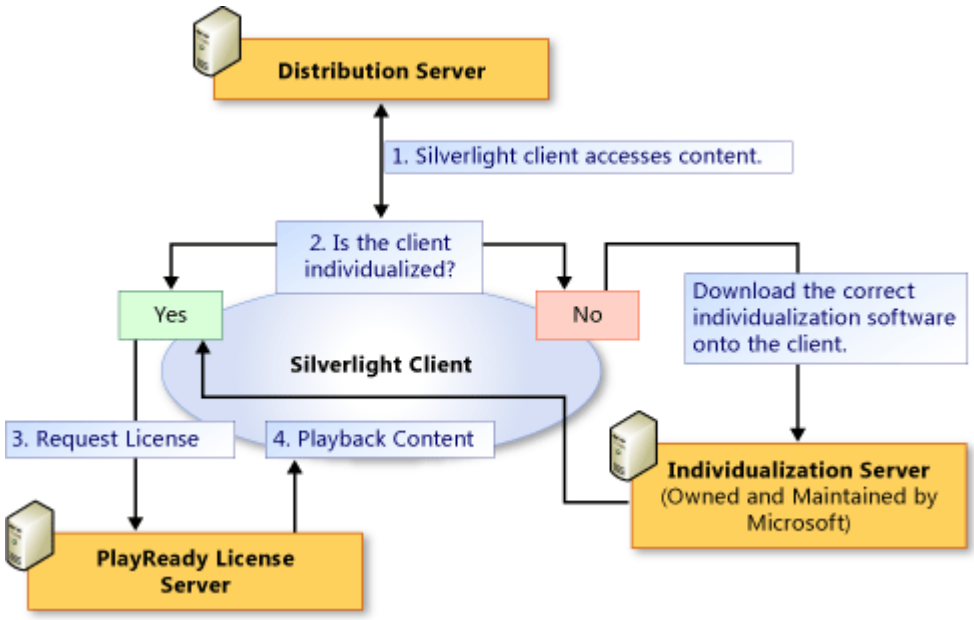
WMDRM je technologií, která vznikla před vznikem Silverlightu a je zde podporována hlavně proto, aby již existující systémy a projekty mohly začít využívat

na klientské straně technologii Silverlight bez toho, aby musely měnit technologii ochrany obsahu nebo dokonce obsah samotný. Takže již vytvořený a zašifrovaný obsah je možné přehrávat například ve Windows Media Playeru, ale nově také v Silverlightu.

Na druhou stranu Silverlight DRM je technologie, která vznikla pro Silverlight a přináší oproti WMDRM různé výhody specifické pro Silverlight. Kromě toho přináší také novější šifrování – Advanced Encryption Standard (AES). Hlavní výhodou u této technologie je, že nemusí být využito pouze klasické streamování, ale může být využito i metody progressive downloadu.

A jak DRM funguje? V tomto případě je pro vysvětlení nejlepší použít obrázků.

Fungování Silverlight DRM

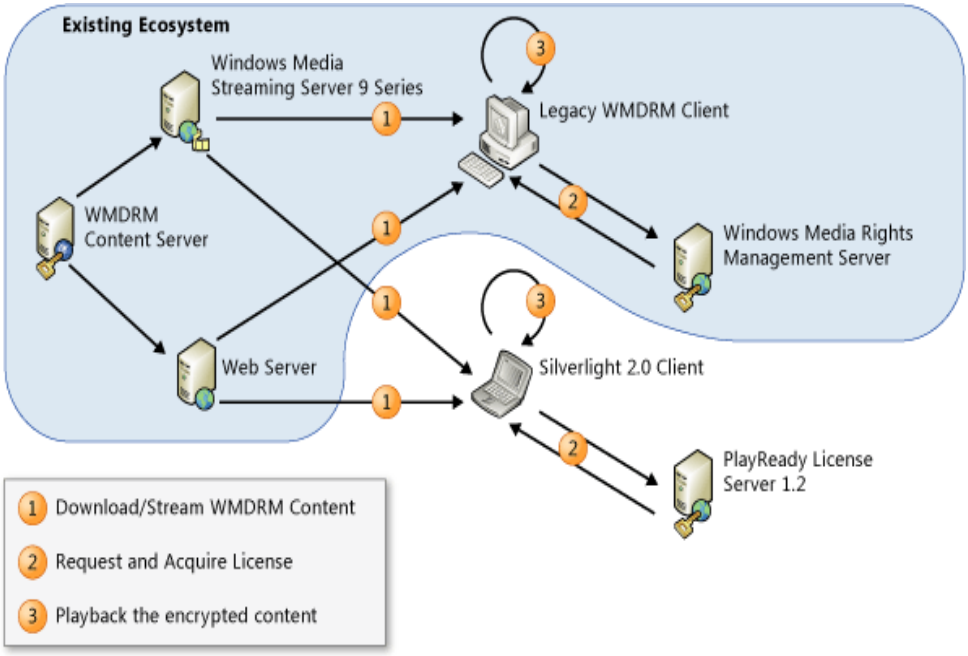


Obrázek 1: Schéma Silverlight DRM (15)

Popis jednotlivých kroků (15):

1) Silverlight Client Accesses Content = Přístup Silverlight klienta k obsahu
   Koncový uživatel se v Silverlight aplikaci pokouší přehrát nějaký DRM chráněný obsah, který je uložen na distribučním serveru. Klientská Silverlight aplikace stáhne zašifrovaný obsah (popřípadě začne přenášet streamovaný obsah) a také hlavičku.
2) Is the Client Individualized? = Je klientská aplikace individualizovaná?
   Předtím než Silverlight bude od licenčního serveru požadovat licenci pro dešifrování obsahu, musí nejprve určit, zda je na uživatelově počítači nainstalovaný příslušný DRM software. Tento software se nazývá individualizační komponenta a je klientskou částí DRM, která je vyžadována předtím než může být přehrán jakýkoliv šifrovaný obsah. Individualizační komponenta umožňuje klientskému počítači požadovat a používat DRM licenci a také chránit citlivá data při dešifrovacím procesu. V případě, že individualizační komponenta není na klientském počítači nainstalována, klient požaduje tuto komponentu z Microsoft Individualization Service. Proces získávání individualizační komponenty se nazývá „individualizace“ (Individualization (15)). Po úspěšné individualizaci je komponenta nainstalovaná, a dokud bude tato komponenta aktuální (než bude vydána nová verze) nebude třeba tento proces opakovat.
3) Silverlight Requests License = Silverlight žádá o licenci
   Když je na klientském počítači nainstalována správná individualizační komponenta, je tento počítač připraven přehrát DRM chráněný obsah. Teď Silverlight klient může kontaktovat PlayReady License Server kvůli získání licence. Když licenční server schválí žádost, vydá licenci, kterou potom klientská aplikace využije pro dešifrování konkrétního souboru. Po dešifrování může být obsah přehrán.

Fungování Windows Media DRM



Obrázek 2: Schéma Windows Media DRM a k němu připojeného Silverlight klienta

Na obrázku je vidět existující projekt, kde se pracuje s WMDRM šifrovaným obsahem a ke kterému se připojuje klientská Silverlight aplikace. Silverlight aplikace vyžaduje PlayReady License Server, odkud získává licence. Z původního projektu Silverlight klient využívá pouze distribuční servery a zašifrovaný obsah, který poskytují.

Záložky

Silverlight, konkrétně ovládací prvek MediaElement nabízí možnost práce se záložkami, dalo by se říci kapitolami (podobné jako na DVD). V Silverlightu se tyto záložky, příslušná třída, nazývá TimelineMarker, v češtině by se dalo říci časová značka.

Se záložkami se pracuje dvěma způsoby. Buď jsou záložky přidány do hlavičky video souboru při kódování videa a MediaElement pracuje s kolekcí načtenou při načtení videa, nebo se dá se záložkami pracovat programově. První způsob je jednoduchý a záložky musí vytvořit poskytovatel obsahu za použití softwaru, který umí video kódovat společně se záložkami. Druhý způsob vyžaduje zapojení programátora a jednou z možností je například XML soubor od poskytovatele videa,

který se programově převede na záložky. To dává autorům a poskytovatelům video obsahu nezávislost na speciálním softwaru schopném kódovat video se záložkami.

Využití záložek je široké a zajímavé. Záložky se dají využít například u výukových videí, kdy se v určitém správném okamžiku zobrazí například doplňující informace, zobrazí se odkazy na doplňující studijní materiály, zobrazí se například dokument, o kterém se ve videu právě hovoří atd. Ovšem jiný příklad je, jako u DVD, rozdělení delšího videa do kapitol pro přehlednost a rychlejší orientaci. Scénářů jak se dají záložky využít je velké množství a představivosti se meze nekladou.

Tato možnost umožňuje mnohem větší interaktivitu a pro uživatele (diváka) nabízí různé nové pohledy na výuku, sledování streamovaného filmu atd. Tato interaktivita a široká využitelnost je dána tím, že při přehrávání videa v MediaElementu jsou záložky kontrolovány a při dosažení záložky je vyvolána událost MarkerReached, která může být obsloužena v závislosti na požadavcích. Každá záložka, instance třídy TimelineMarker, může mít vlastní textový titulek (vlastnost TimelineMarker.Text) a také typ (vlastnost TimelineMarker.Type) reprezentovaný řetězcem. Tyto vlastnosti mohou být využity pro odlišení jednotlivých záložek a programátor má možnost obsloužit událost MarkerReached úplně odlišně pro každou záložku. To umožňuje zobrazovat konkrétní pokročilý obsah určený pro aktuální záložku.

Server-side Playlist – SSPL

Server-side Playlist (dále jen SSPL) jsou playlisty na straně serveru. To znamená, že je tvoří a určuje poskytovatel obsahu a poskytují mu kontrolu nad tím, jaké soubory (audio i video) si uživatel přehraje a v jakém pořadí si soubory přehraje. Pro uživatele se tato sekvence souborů chová jako jeden soubor, protože se další soubor přehrává automaticky bez toho, aby uživatel musel cokoliv udělat.

Tyto SSPL mohou být vytvářeny nejen staticky, ale i dynamicky, což umožňuje poskytovateli obsahu další možnosti. Omezení SSPL je, že mohou být využity pouze při klasickém streamingu, tedy i živém i na vyžádání, nemohou být využity při progressive download.

Používání SSPL se dá využít pro různé situace a přináší výhody hlavně poskytovatelům obsahu. SSPL se dají využít například pro nastavení a omezení přehrávání určitých souborů. To znamená, že se některé soubory dají v určitém okamžiku znepřístupnit. Využití je tady opět široké a záleží na požadavcích, jak se SSPL využijí.

Ke konfiguraci a vytváření SSPL se využívá konfigurační XML soubor s příponou wsx. Ten je pro použití SSPL potřeba vytvořit na serveru. Tento soubor se potom nastaví MediaElementu jako zdroj a SSPL může být přehráván. Pro používání SSPL v Silverlightu platí jistá omezení, takže nejsou podporovány všechny funkce SSPL, proto je potřeba pracovat s dokumentací (12).

Multicast

V oblasti streamování audia/videa a přehrávání streamovaného obsahu je multicast velice důležitou technologií, která ovšem zatím není tak rozšířená tak, jak by si zasloužila. Největší výhodu tato technologie přináší poskytovatelům obsahu, protože podstatně snižuje zátěž serveru a připojení serveru k internetu. Multicast funguje tak, že namísto toho, aby se pro každého klienta vytvářelo nové připojení, se data vysílají (odcházejí ze serveru) pouze jednou a ta samá data jdou poté k několika klientům. Data se kopírují až tam, kde je to nutné a tím se snižuje zatížení sítě.

Například u internetového rádia nebo televize, najednou se připojují tisíce klientů. Připojeným klientům se ve stejném okamžiku posílají stejná data. V případě, kdy není využito možností multicastu, se stejná data ze serveru musí odeslat v jednom okamžiku tisíckrát (každému klientovi zvlášť). Oproti tomu pokud se využije multicast, je třeba data ze serveru posílat pouze jednou. A to znamená značné snížení zátěže serveru i sítě.

Z výše popsaných výhod vyplývá logický požadavek poskytovatelů streamovaného obsahu na to, aby Silverlight také přehrával multicast vysílání. Stejné požadavky měli klienti společnosti Qumu a tak ve spolupráci této společnosti se společností Microsoft vznikl plug-in pro Silverlight, který multicast umožňuje. Tento multicast plug-in, který se nazývá Starlight je ke stažení (http://projectstarlight.codeplex.com) zcela zdarma včetně zdrojových kódů a je

odpovědí na požadavky poskytovatelů obsahu streamovaného obsahu. Tento plug-in umožňuje přehrávat multicast streamování z Windows Media Services (WMS). Další z výhod je i to, že pokud poskytovatel obsahu již využívá multicast s WMS ve svých projektech, může bez jediného problému přejít na Silverlight přehrávače, protože není nutná žádná úprava pro stávající síť, ani server.

4.1.4 Zdroje pro MediaElement

V předchozí kapitole bylo řečeno, že hlavní třídou pro práci s videem, audiem, takže veškerým streamovaným obsahem je třída MediaElement. Také bylo řečeno, že MediaElement a Silverlight umožňuje přehrávání různých formátů videa a také podporuje různé způsoby pro přístup k přehrávanému obsahu. V souvislosti s těmito metodami přístupu k obsahu bylo výše vysvětleno, že MediaElement v závislosti na zadaném zdroji sám rozhoduje, jakou metodu zvolí.

Zatím ale nebylo vysvětleno, jak se tento zdroj nastavuje a jaké možnosti v tomto ohledu Silverlight nabízí. MediaElement nabízí tři možnosti jak nastavit zdroj přehrávání:

- Nastavení vlastnosti Source na URI přehrávaného souboru.
- Metoda SetSource s parametrem typu Stream.
- Metoda SetSource s parametrem typu MediaStreamSource.

Nastavení vlastnosti Source

Základním způsobem je nastavování vlastnosti Source na URI (třída Uri) přehrávaného souboru. Jedná se o způsob, který je využit například ve Windows Media Playeru. Uživatelé jsou na tento způsob nastavování zdroje zvyklí. Takže pokud uživatel používá tento přehrávač, stačí pouze zadat URI cestu k souboru a MediaElement tento soubor začne přehrávat. Tento základní způsob dává programátorům možnost nastavit zdroj přímo v XAML kódu, takže přehrávač může být naprogramovaný zcela bez nějakého kódu na pozadí (code-behind) – to umožňuje plně využít veškerých výhod XAML. Tento zdroj (vlastnost Source) může být nastaven přímo v XAML, v kódu na pozadí (C# nebo VB.NET) a může být nastaven pevně nebo může být uživateli umožněno tento zdroj měnit a nastavovat vlastní zdroj.

Do vlastnosti Source je možné vkládat i relativní odkazy. Tyto relativní odkazy mají defaultně nastaveno startovní umístění. Toto defaultní umístění je ve stejném místě, kde je umístěný XAP soubor, tedy Silverlight aplikace s MediaElementem. Samozřejmostí jsou i absolutní odkazy, které mohou odkazovat i na umístění na jiné webové stránce než na které je hostována Silverlight aplikace, i na jiném serveru. U

těchto odkazů, URI cest k souboru, je možné využít nejen protokolu http, ale je možné využít i URI začínající na mms:, rtsp: nebo rtspt:. A podle těchto předpon MediaElement vybere primární metodu doručování obsahu.

Pokud URI cesta k souboru není validní nebo pokud soubor, na který odkazuje zadané URI, není validní nebo soubor není v podporovaném formátu, tak je vyvolána událost MediaFailed. Tato událost ovšem není vyvolána, pokud neuspěje nastavení Source jako cross-domain URI, to znamená v případě, že je špatný zdroj v jiné doméně než Silverlight aplikace. Při nastavování URI jako vlastnosti Source MediaElementu musí být využívána pouze lomítka (/) a nesmí být využívána zpětná lomítka (\).

Metoda SetSource s parametrem typu Stream

Dalším způsobem, jak nastavit MediaElementu zdrojový audio/video soubor je využít metodu SetSource, která nabízí 2 možnosti nastavení zdroje. První možností je předat metodě SetSource parametr typu Stream. Tento přístup je vhodný tam, kde není možné nastavit přímo URI zdrojového souboru. Dále je vhodný tam, kde je audio/video soubor umístěn přímo v Silverlight aplikaci, takže v případě, že je součástí XAP souboru. Třetí možností, kde se použití tohoto způsobu nabízí je ten, kdy zdroj je dostupný jedině jako Stream. Společně s tímto způsobem se velice často využívá třídy WebClient, která dokáže přistupovat k souborům na zadané URI adrese, jako k instancím tříd odvozených z abstraktní třídy Stream (29).

Metoda SetSource s parametrem typu MediaStreamSource

Poslední možností, jak nastavit zdroj MediaElementu je využít opět metodu SetSource, ale jako parametr jí předat parametr typu MediaStreamSource. Tento způsob existuje pouze pro Silverlight a umožňuje používat i nepodporované formáty. Jak už bylo řečeno v kapitole o podporovaných kontejnerech a kodecích, tak Silverlight v současné verzi nabízí určitou nezávislost na podporovaných kontejnerech a kodecích. Tato nezávislost je poskytována díky abstraktní třídě MediaStreamSource a s tím související metodou SetSource, které instanci třídy odvozené z této třídy předáváme jako parametr.

Tímto způsobem MediaElement dokáže přehrávat i jiné než podporované formáty videa a audia. Pro příklad tato možnost přidává funkcionalitu pro přehrávání nepodporovaného formátu audia WAV (27, 28).

Protože MediaStreamSource je relativně novou vlastností a hlavně odlišností Silverlightu, tak práce s touto třídou je popsána podrobněji dále.

4.1.5 Třída MediaStreamSource

Třída MediaStreamSource již byla zmíněna jako třída, která poskytuje nezávislost na podporovaných kodecích a kontejnerech. Pro příklad bylo výše uvedeno, že pomocí této třídy je možné přidat do Silverlight aplikace podporu WAV souborů (27, 28), které nepatří mezi defaultně podporované formáty.

To ovšem není jediná možnost, jak se tato třída dá využít. Možnosti použití jsou široké, například se dá video vytvářet programově, takže je generováno náhodně, podle nějakého vzorce atd., záleží na konkrétním scénáři a požadavcích. Použití není omezeno ani na existující obsah.

Navíc také poskytuje nové možnosti při použití různých verzí zvuku (jazykové verze, komentáře), různých verzí videa (například jiný úhel pohledu, jiná kvalita videa atd.). Třída MediaStreamSource rozšiřuje možnosti MediaElementu a Silverlightu.

Tato třída poskytuje všechny funkce bez interakce s uživatelem, takže uživatel nemusí cokoliv dalšího instalovat, nastavovat nebo registrovat. Vše potřebné je v XAP souboru a vše co je vyžadováno je nainstalovaný Silverlight. To znamená, že využití funkcionality třídy MediaStreamSource je pro uživatele zcela bez potíží a může si užívat výhod a možností, které třída MediaStreamSource nabízí.

Další možností, kterou poskytuje využití třídy MediaStreamSource je vytvoření a použití vlastní transportní metody. Takže Silverlight není omezen ani na 2 hlavní metody přenášení obsahu ze serveru – klasický streaming a progressive download. Díky tomu může Silverlight poskytovat vlastní implementaci adaptive streamingu, který je založen na využití MediaStreamSource. Tato implementace adaptivního

streamingu se nazývá Smooth Streaming (objevuje se i delší pojmenování – IIS Smooth Streaming (41)). O Smooth Streamingu více později.

MediaStreamSource je abstraktní třída a poskytuje programátorovi přístup i k nepodporovaným formátům, které se převádí na formát, kterému MediaElement rozumí, viz podporované formáty. Abstraktní třídou je MediaStreamSource proto, že obsahuje abstraktní metody, které programátor musí implementovat pro konkrétní scénáře. To znamená, že by nebylo možné vytvořit tyto metody tak, aby byly použitelné pro všechny případy. To, že je tato třída abstraktní, poskytuje programátorům nezávislost a použitelnost pro různé formáty.

Jak se instance tříd odvozených ze třídy MediaStreamSource používají a nastavují jako zdroje pro MediaElement pomocí metody SetSource třídy MediaElement, bylo řečeno a vysvětleno v předchozí kapitole.

Abstraktní metody třídy MediaStreamSource

Abstraktní třída by nebyla abstraktní třídou, pokud by neobsahovala abstraktní metody, které musí implementovat programátor, který tvoří odvozenou třídu. Tyto metody jsou volány MediaElementem. Třída MediaStreamSource obsahuje tyto abstraktní metody:

- CloseMedia()
- GetDiagnosticAsync(MediaStreamSourceDiagnosticKind)
- GetSampleAsync(MediaStreamType)
- OpenMediaAsync()
- SeekAsync(long)
- SwitchMediaStreamAsync(MediaStreamDescription)

Metoda CloseMedia()

Tato metoda je MediaElementem volána, když je přehrávání normálně zastaveno (konec přehrávání nebo zastavení uživatelem) nebo když dojde k chybě a přerušení je zastaveno v důsledku chyby.

V této metodě by měly být uzavřeny otevřené streamy a měly být nastaveny defaultní hodnoty, jako třeba aktuální pozice přehrávání by měla být nastavena na počáteční. Třída by měla být celkově uklizena a připravena na nové přehrávání.

Ukázka:

```
protected override void CloseMedia()
{
      actualTimeStamp = 0;
      sampleStream.Close();
      sampleStream.Dispose();

}
```

Metoda GetDiagnosticAsync(MediaStreamSourceDiagnosticKind)

Metoda je volána MediaElementem, když MediaElement požaduje jistý druh diagnostických informací. Druh požadovaných informací udává parametr, který je výčtového typu MediaStreamSourceDiagnosticKind. Tento typ popisuje typ diagnostických informací používaných médiem. Tento typ může nabývat hodnot BufferLevelInMilliseconds a BufferLevelInBytes, význam těchto hodnot je zřejmý z jejich názvu.

V této metodě by třída odvozená od třídy MediaStreamSource měla informovat MediaElement o hodnotě, která k požadovanému údaji náleží. Informování se provádí pomocí volání metody ReportGetDiagnosticCompleted, které se jako parametry předávají nejprve typ informací (stejný jako byl předaný metodě GetDiagnosticAsync) a poté hodnota k této informaci, která je číselného typu long.

GetSampleAsync(MediaStreamType)

Tuto metodu volá MediaElement vždy, když chce získat ze zdroje (třídy odvozené z MediaStreamSource) další vzorek (v anglické literatuře se objevuje pojem sample (42)), který poté zobrazí (přehraje). Tento vzorek může být audio nebo video, což udává parametr předávaný této metodě. Tento parametr je výčtového typu MediaStreamType a nabývá hodnot Audio, Video nebo Script (ten ale není ve třídě MediaStreamSource podporován (43)).

V této metodě můžeme informovat MediaElement pomocí metody ReportGetSampleProgress o tom, že vzorek není připraven a můžeme informovat o průběhu vytváření a připravování vzorku. Metodě ReportGetSampleProgress se jako parametr předává hodnota od 0 do 1, která udává stav zpracování.

Důležitější je ale pomocí metody ReportGetSampleCompleted informovat MediaElement o tom, že vzorek byl vytvořen a předat tento vzorek jako parametr. Vzorek, který je odesílán MediaElementu, je typu MediaStreamSample a obsahuje nejen data, ale také popis dat. Třída MediaStreamSample popisuje vzorek dostatečně podrobně tak, aby MediaElement dokázal zobrazit tento vzorek, k tomu se využívá vlastností:

- Attributes – kolekce atributů, které určují, jak má MediaElement zobrazit vzorek
- Count – počet bajtů ve streamu, které reprezentují aktuální vzorek
- MediaStreamDescription – popisuje stream, ze kterého vzorek pochází
- Offset – posun od začátku streamu ke skutečnému začátku vzorku, například pro přeskočení hlavičky
- Stream – stream (proud) dat, který reprezentuje vzorek
- Timestamp – čas, který vyjadřuje časovou pozici, udává se ve stovkách nanosekund

Pokud je metodě ReportGetSampleCompleted jako parametr předána hodnota null, MediaElement ukončí přehrávání.

Ukázka:

```
protected override void GetSampleAsync(MediaStreamType mediaStreamType)
 {
     if (mediaStreamType == MediaStreamType.Video)
     {
         Random r = new Random();
         for (int i = 0; i < sampleBufferSize; i++)
         {
             sampleData[i] = (byte)r.Next(0, 255);
         }

         if (sampleStream != null)
         {
             sampleStream.Close();
             sampleStream.Dispose();
         }
         sampleStream = new MemoryStream();
         sampleStream.Write(sampleData, 0, sampleBufferSize);

         MediaStreamSample msSample = new MediaStreamSample(
             videoDescription,
             sampleStream,
             0,
             sampleBufferSize,
             actualTimeStamp,
             emptySampleDict);

         actualTimeStamp += sampleTime;

         ReportGetSampleCompleted(msSample);
     }
 }
```

OpenMediaAsync()

MediaElement tuto metodu volá poté, co je mu metodou SetSource přiřazena jako zdroj instance třídy odvozené ze třídy MediaStreamSource. Před přehráváním zadaného zdroje. MediaElement od zdroje požaduje skrze tuto metodu metadata potřebná pro start přehrávání.

V této metodě je potřeba pomocí metody ReportOpenMediaCompleted předat MediaElementu nejprve slovník atributů (například informace o době trvání nebo informace o tom, zda se dá v přehrávání přesunovat v čase) a poté kolekce dostupných audio a video streamů (různé kvality, různé úhly pohledu, různé jazykové verze zvuku). Tato kolekce obsahuje instance typu MediaStreamDescription, které obsahují informaci o tom, zda se jedná o audio nebo video stream a také kolekci atributů (pro video stream například: formát dat, výška, šířka atd.).

Ukázka:

```
protected override void OpenMediaAsync()
{
    Dictionary<MediaSourceAttributesKeys, string> sourceAttributes =
        new Dictionary<MediaSourceAttributesKeys, string>();
    sourceAttributes[MediaSourceAttributesKeys.Duration] =
          0.ToString();
    sourceAttributes[MediaSourceAttributesKeys.CanSeek] =
    false.ToString();

    Dictionary<MediaStreamAttributeKeys, string> streamAttributes =
        new Dictionary<MediaStreamAttributeKeys, string>();
    streamAttributes[MediaStreamAttributeKeys.VideoFourCC] = "RGBA";
    streamAttributes[MediaStreamAttributeKeys.Width] =
    videoWidth.ToString();
    streamAttributes[MediaStreamAttributeKeys.Height] =
    videoHeight.ToString();

    videoDescription =
    new MediaStreamDescription(MediaStreamType.Video, streamAttributes);

    availableStreams = new List<MediaStreamDescription>();
    availableStreams.Add(videoDescription);

    ReportOpenMediaCompleted(sourceAttributes, availableStreams);
}
```

SeekAsync(long)

MediaElement volá tuto metodu v případě, že se uživatel chce v přehrávání posunout na nějaký čas. Jako parametr je předáno číslo typu long, které udává posun oproti počáteční pozici, (pozici, na kterou se uživatel chce přesunout).

V této metodě je potřeba nastavit, aby příští volání metody GetSampleAsync vracelo snímek na požadované pozici. Po úspěšném dokončení je potřeba informovat MediaElement pomocí metody ReportSeekCompleted, této metodě předáváme jako parametr číselnou reprezentaci času, kam jsme se posunuli.

SwitchMediaStreamAsync(MediaStreamDescription)

MediaElement volá tuto metodu, pokud se rozhodl přehrávat jiný stream. Parametr předávaný této metodě je stream, který chce MediaElement přehrávat. Tento stream je reprezentován instancí třídy MediaStreamDescription.

V této metodě je nutné nastavit, aby příští volání metody GetSampleAsync vracelo snímek z požadovaného streamu. To znamená na stejné pozici, ale například jinou jazykovou verzi atd.

Další informace k abstraktním metodám třídy MediaStreamSource

Tyto metody, slouží k asynchronní komunikaci mezi MediaElementem a zdrojem, který je reprezentován instancí třídy odvozené ze třídy MediaStreamSource. Tyto abstraktní metody jsou volány MediaElementem, který tento zdroj přehrává. Odpovědi jsou realizovány prostřednictvím volání metod Report…, které jsou částečně popsány u metod, na které odpovídají.

U popisu některých metod se objevuje ukázkový zdrojový kód, který ukazuje možnou implementaci těchto abstraktních metod. Konkrétně se jedná o metody CloseMedia(), GetSampleAsync(MediaStreamType) a OpenMediaAsync(). Pro nejjednodušší MediaStreamSource je možné pracovat pouze s posledními dvěma jmenovanými metodami. Stačí pouze otevřít a nastavit zdroj a poté posílat MediaElementu snímky. Tyto ukázkové metody pocházejí z jednoduché třídy odvozené ze třídy MediaStreamSource, která automaticky generuje náhodné snímky a tyto snímky posílá MediaElementu a vytváří dojem video šumu. Okomentovaný zdrojový kód této třídy je k dispozici v přílohách k práci, viz soubor VideoGenMediaStreamSource.cs. Instanci této třídy stačí nastavit jako zdroj MediaElementu a spustit přehrávání a MediaElement začne přehrávat generované video – „šum“.

4.2 Smooth Streaming

Označení Smooth Streaming (oficiální název IIS Smooth Streaming (45)) se v předchozích kapitolách objevilo několikrát. V prvním případě to byla kapitola věnovaná Adaptive streamingu, ve druhém případě to bylo v kapitole věnované třídě MediaStreamSource. Smooth Streaming je implementací adaptive streamingu založeném na protokolu http od společnosti Microsoft a jedná se o technologii určenou pro Silverlight (44). Smooth Streaming je postaven na MediaStreamSource API. Všechna funkcionalita je poskytována díky .NET kódu, například vybírání správného souboru (datového toku), stahování fragmentů přes http atd.

Smooth Streaming je technologie, která přizpůsobuje datový tok tak, aby přehrávání začalo okamžitě po zapnutí, aby okamžitě po přesunutí na určitou pozici v čase začalo přehrávání na vybrané pozici a aby byla přehrávána nejlepší možná kvalita (nejvyšší datový tok), kterou umožňují aktuální podmínky sítě a procesoru. Tyto podmínky jsou kontrolovány v každém okamžiku přehrávání a přehrávání je závislé na aktuálním stavu sítě a procesoru.

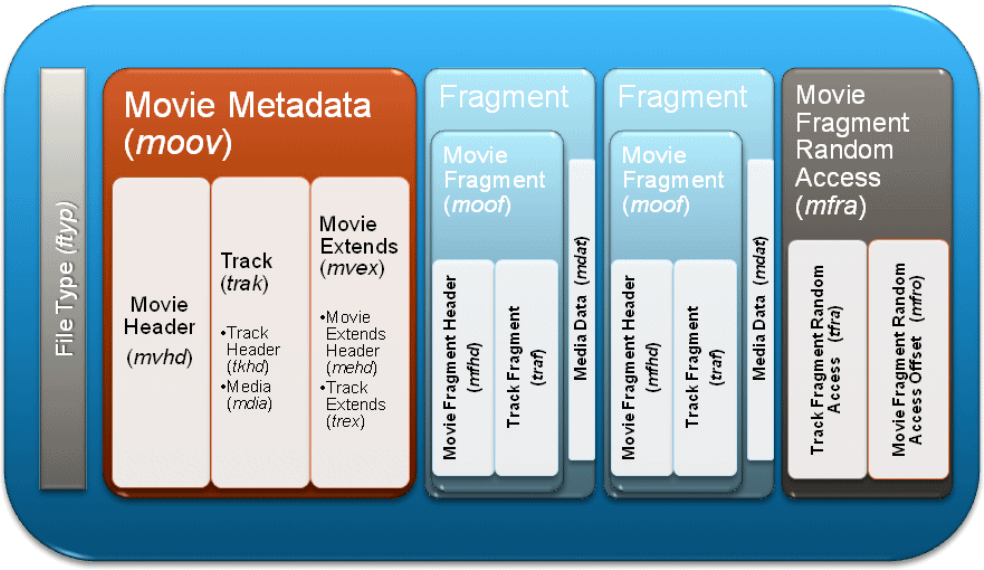
Všechny výše zmíněné možnosti Smooth Streamingu jsou zaměřeny na co nejlepší zážitek uživatele z přehrávání. Uživatel, který si chce přehrát video v HD kvalitě, nechce čekat několik vteřin, až se do bufferu stáhne alespoň nějaká část souboru, kterou může přehrávač přehrát. Uživatel chce spustit video a okamžitě ho sledovat. Toto Smooth Streaming umožňuje.

4.2.1 Formát souborů

Jak už bylo řečeno, Smooth Streaming využívá speciální formát souborů, který je vyžadován k tomu, aby bylo možné plynule přecházet z přehrávání jednoho souboru na přehrávání jiného souboru. Smooth Streaming totiž funguje tak, že nepřenáší celé soubory, ale pouze fragmenty souborů. Kdyby byly tyto fragmenty na serveru umístěny jako jednotlivé soubory, tak by to znamenalo velikou zátěž na vyhledávání a celkově na správu takovéhoto množství jednotlivých souborů. Fragmenty, které se přenáší ze serveru ke klientovi, mají standardně délku dvě vteřiny (47) a tak by minuta videa uloženého v 6 datových tocích znamenala 30 souborů na jeden datový tok, celkově 180 jednotlivých souborů na minutu videa! To je nežádoucí a tak byl vytvořen zvláštní formát souborů, kde existuje jeden soubor pro datový tok a ten není rozdělen fyzicky na fragmenty, ale fragmenty obsahuje uvnitř, stejně jako metadata, která popisují rozložení fragmentů v souboru.

Formát souborů pro Smooth Streaming je založený na MP4 formátu (ISO/IEC 14496-12 ISO Base Media File Format) (47). MP4 formát je totiž oproti například ASF formátu menší, v .NET je lépe parsovatelný a navíc je formát MP4 založen na velice využívaném kodeku H.264 (47). Takovýto soubor je na serveru uložen pro každý datový tok jeden a každý má plnou délku videa. Takovýto MP4 soubor je

ovšem zabalený v tzv. boxu, který kromě tohoto MP4 souboru obsahuje také zmíněná metadata. Tyto MP4 soubory, které jsou ve skutečnosti organizovány jako fragmenty videa, se nazývají „Fragmented MP4“ soubory nebo „(f)MP4“ (47). Ukázka rozložení tohoto formátu viz obrázek 3. Každý tento soubor začíná metadaty souboru, která popisují soubor. Tato metadata se označují Movie Metadata nebo zkráceně moov, viz obrázek 3. Na obrázku jsou pro zjednodušení vyobrazeny pouze dva fragmenty, ale ve skutečnosti tyto soubory obsahují velký počet těchto fragmentů. Soubor je ukončen indexovacím polem (mfra), které usnadňuje a zpřesňuje přehrávání na určité pozici. Tento formát je využíván na straně serveru a jedná se prakticky o úložiště fragmentů.

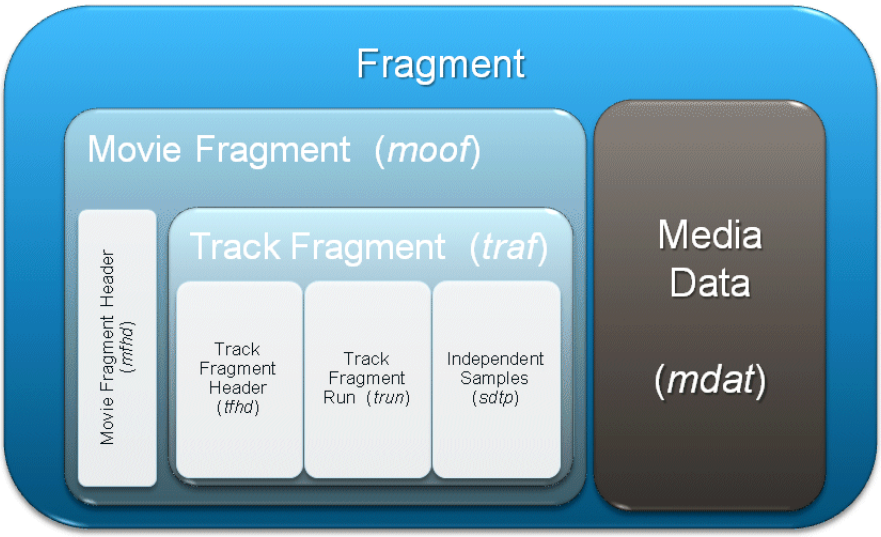


Obrázek 3: Smooth Streaming - formát souborů (47)

Tyto soubory ale nejsou odesílány ze serveru klientovi. Klientovi jsou odesílány pouze fragmenty, které obsahují část souboru. Tato část souboru musí mít stejnou dobu trvání, stejnou dobu začátku i konce, jako fragmenty z jiného souboru (stejného souboru s jiným datovým tokem). To je důležité, aby mohly být fragmenty přehrávány nezávisle na sobě a přehrávání bylo plynulé. Fragmenty se od sebe odlišují hlavně velikostí, protože délka i pozice jsou stejné ve všech souborech.

Tyto fragmenty obsahují přesnější metadata na úrovni fragmentů. Tato metadata jsou označována Fragment-level Metadata (moof). A fragment zároveň obsahuje data, která se budou přehrávat. Tato data jsou tím nejdůležitějším, ale musí být

popsána metadaty a obsažena ve fragmentech, aby bylo snadné je na serveru vyhledat a odeslat přehrávači pouze požadovanou část dat. Vnitřní schéma fragmentu je zobrazeno na obrázku 4.



Obrázek 4: Schéma fragmentu (47)

Zatím byla popsána jen vnitřní organizace souborů, které obsahují data přehrávaná Silverlight přehrávačem na straně klienta. Pro tyto soubory se využívají koncovky:

- ismv – video a audio obsah – IIS Smooth Streaming Video
- isma – pouze audio obsah – IIS Smooth Streaming Audio

Ovšem toto nejsou jediné soubory, které pro svou správnou funkci vyžaduje Smooth Streaming. Vyžadovány jsou také soubory, které popisují dostupné soubory (dostupné datové toky), záložky a další doplňující informace.

Klasicky Smooth Streaming obsah musí obsahovat tyto soubory:

- Fragmented MP4 – výše popisované soubory, které jsou určené pro video a audio soubory
- Server Manifest – koncovka ism – popisuje vztahy mezi ismv soubory, tento soubor je využíván pouze serverem
- Client Manifest – koncovka ismc – popisuje použité kodeky, dostupné streamy, kvality, rozlišení, záložky atd., jedná se o první soubor, který si klient stahuje

Manifest soubory (.ism a .ismc) jsou založeny na XML (konkrétně SMIL 2.0 XML). To, že jsou tyto konfigurační soubory založeny na XML, umožňuje snadné rozšiřování možností. Současný Smooth Streaming formát je připraven na možnosti:

- Různé kodeky – VC-1, H.264, WMA, AAC…
- Vícejazyčné audio stopy – podpora několika audio stop u jedné video stopy
- Alternativní video i audio stopy – záběry z jiné kamery, komentáře tvůrců a další využití
- Skriptování, kapitoly, záložky
- Komprimování Client Manifest souboru – kvůli rychlejšímu stažení
- Živé kódování a streaming – IIS Live Smooth Streaming (48)

Přehrávač, který má přehrávat Smooth Streaming obsah, musí mít nastaven jako zdroj cestu ke Client Manifest souboru. První věcí, kterou Silverlight přehrávač udělá je, že tento konfigurační soubor stáhne. Z tohoto souboru přehrávač zjistí, jaký kodek byl použit pro komprimaci obsahu, jaké jsou rozměry přehrávaného videa. Silverlight klient v závislosti na těchto informacích použije správný dekodér. Z tohoto souboru také Silverlight klient zjistí dostupné datové toky a rozlišení, jaké jsou fragmenty dat (doba trvání, začátek, konec). Takto se Silverlight klient připraví na přehrávání Smooth Streaming obsahu. Z tohoto popisuje je patrná jistá shoda s popisem metody OpenMediaAsync ze třídy MediaStreamSource na které je Smooth Streaming založen. Nejprve se zjistí a nastaví kodek a formát dat a poté se vytvoří kolekce dostupných streamů.

Poté Silverlight klient vybírá z dostupných datových toků ten, který je právě nejvhodnější a když vybere správný, požádá server o příslušný fragment. Po přenesení je fragment přehrán a proces výběru a přenosu se opakuje. Tady se opět objevují shodné rysy se třídou MediaStreamSource a konkrétně s metodou GetSampleAsync.

4.2.2 Vývoj přehrávače pro Smooth Streaming

Pokud chce programátor vytvořit přehrávač, který podporuje přehrávání Smooth Streaming obsahu, má několik možností. Základní možností, jak vytvořit Smooth

Streaming přehrávač je využití šablon poskytovaných aplikací Expression Encoder 3 (nebo Expression Encoder 2 s doplňkem pro Smooth Streaming). V této aplikaci je možné video přímo kódovat do správného formátu a to dokonce do několika datových toků. Aplikace poskytuje všechny potřebné funkce pro vytvoření Smooth Streamingu obsahu. Tato aplikace také nabízí šablony pro přehrávače, které umí přehrávat Smooth Streaming obsah, stačí pouze vybrat šablonu, nastavit požadovaná rozlišení, datové toky a další nastavení pro kódování obsahu a Expression Encoder vytvoří Silverlight přehrávač, který přehrává právě kódovaný Smooth Streaming obsah. K těmto šablonám jsou k dispozici také zdrojové kódy, takže je možné vytvořit přehrávač odvozený z těchto šablon.

Další možností je využít již vytvořeného přehrávače, který je dostupný na internetu. Například přehrávač z projektu Silverlight Media Framework (49), ke kterému jsou dostupné zdrojové kódy a také výuková videa, které vývojáři pomohou při práci s tímto projektem.

Třetí možností je vytvořit celý přehrávač zcela od začátku. K tomu je potřeba vědět, jak naprogramovat klasický přehrávač, který využívá prvku MediaElement a jeho metod a vlastností. Jediný rozdíl spočívá v tom, že není možné využívat obyčejný MediaElement, ale je potřeba využít třídu SmoothStreamingMediaElement. Tato třída je obsažena ve jmenném prostoru Microsoft.Web.Media.SmoothStreaming, jež není součástí Silverlightu, ale je dostupné z DLL knihovny Microsoft.Web.Media.SmoothStreaming.dll. Tato knihovna je k dispozici například ve výše zmíněných zdrojových kódech šablon nebo ke stažení na internetu (50). Se třídou SmoothStreamingMediaElement se pracuje podobně jako s třídou MediaElement. Třídě SmoothStreamingMediaElement se zdroj nastavuje také jako URI odkazující na Client Manifest soubor. Tuto URI lze nastavit jako hodnotu vlastnosti Source. Druhou možností je jako zdroj nastavit Stream k tomuto souboru. Takto lze zdroj nastavit pomocí metody SetSource(Stream). Není možné nastavit jako zdroj MediaStreamSource, protože se jedná o implementaci této třídy, takže není možné takto zrušit funkci Smooth Streamingu. Bylo by to zbytečné, pokud někdo chce využít vlastní zdroj založený na MediaStreamSource může využít klasický MediaElement.

5 Ověření vybraných možností Silverlightu

5.1 Úvod do měření

Pro ověření výše popsaných možností technologie Silverlight při přehrávání streamovaného obsahu posloužil experiment, který je popsán v této kapitole.

Tento experiment byl spojen s měřením, které mělo dokázat nebo vyvrátit tvrzení z předchozích kapitol. Tato tvrzení jsou založena na informacích zveřejněných vývojáři technologie Silverlight a také jsou založena na oficiální dokumentaci Silverlightu. Tímto experimentem a měřením byly ověřovány následující funkce:

- Podpora HD videa
- Podpora kodeků H.264 a VC-1
- Podpora GPU
- Out of Browser
- Podpora různých prohlížečů
- Podpora různých metod doručování obsahu

5.1.1 Podpora HD videa

Experiment a měření ověřilo, zda Silverlight opravdu nabízí možnost sledovat video v HD kvalitě. Nejen, je-li možné video v HD kvalitě přehrávat, ale je-li to v současnosti využitelné. Přehrávání videa v této kvalitě je náročné na výpočetní výkon a tak bylo změřeno, jak přehrávání tohoto videa zatěžuje systém.

5.1.2 Podpora kodeků H.264 a VC-1

Spolu s podporou videa v HD kvalitě byla také empiricky ověřena podpora těchto dvou kodeků. Tyto kodeky byly také porovnány z pohledu zatížení systému a bylo experimentem a měřením zjištěno, zda je vhodnější používat kodek H.264 nebo VC-1, když je k přehrávání streamovaného obsahu využita technologie Silverlight.

5.1.3 Podpora GPU

Dále tento experiment ověřil, zda zapnutí GPU akcelerace má vliv na přehrávání streamovaného obsahu a zda dokáže povolení GPU akcelerace snížit zatížení

procesoru. Toto měření bylo prováděno s použitím dvou různě výkonných grafických karet, aby bylo možné určit, zda lepší grafická karta znamená lepší výsledky při zapnuté GPU akceleraci nebo zda jsou výsledky nezávislé na výkonu grafické karty.

5.1.4 Out of Browser

Tento experiment také ověřil, zda při přehrávání v aplikace běžící mimo internetový prohlížeč, stoupá nebo klesá zatížení systému.

5.1.5 Podpora různých prohlížečů

Experimentem byla ověřena také podpora různých prohlížečů. Byly testovány prohlížeče, které aktuálně patří mezi nejčastěji používané prohlížeče (59) – Firefox, Internet Explorer 8 a Google Chrome. Tyto tři nejpoužívanější prohlížeče jsou oficiálně podporovány Silverlightem. Technologie Silverlight je vyvíjena společností Microsoft, stejně jako prohlížeč Internet Explorer a lze předpokládat, že pro tento prohlížeč bude nejlépe optimalizována. Jestli tomu tak doopravdy je, bylo také ověřeno a změřeno v popisovaném experimentu.

5.2 Metodika experimentu

O tom, co bylo v popisovaném experimentu měřeno, pojednává předchozí kapitola. To, jak bylo měření prováděno, jakými nástroji se měřilo a jaké atributy byly měřeny a porovnávány, je popsáno dále.

5.2.1 Testovací video soubory

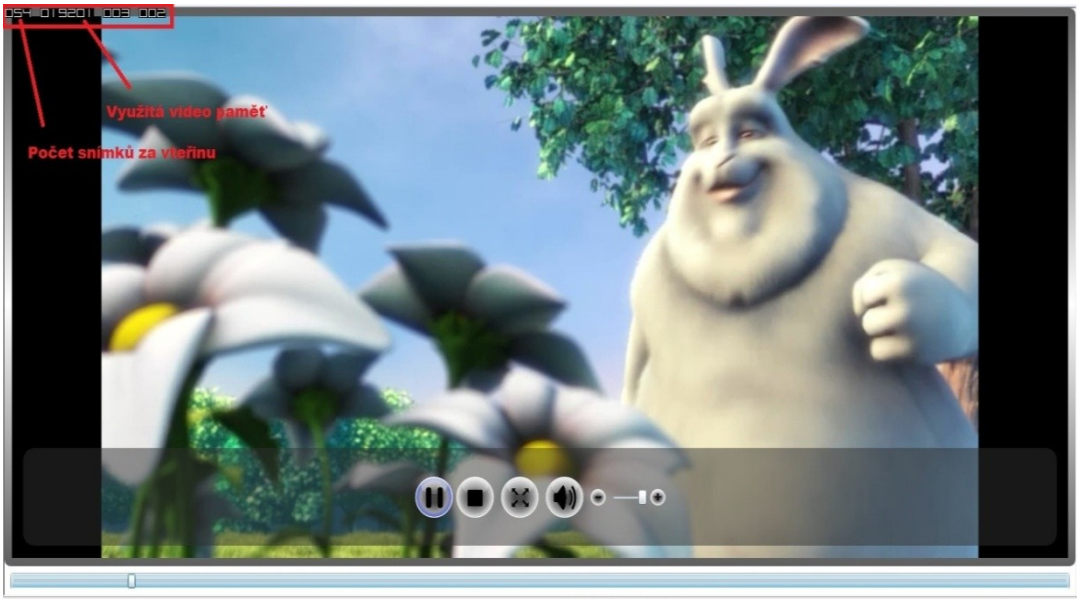
Měření probíhalo na testovacích videích, která byla kódována v deseti různých datových tocích a dvou různých kodecích. Originální video, které bylo využito k vytvoření testovacích video souborů, mělo rozlišení 1920 na 1080 pixelů, kódováno bylo kodekem MPEG-4 V2 s datovým tokem 9341 Kbps.

5.2.2 Měřící nástroje

Testovací videa, na kterých bylo měřeno, byla přehrávána v přehrávači, který je přiložený k této práci včetně zdrojových kódů. Tento přehrávač vznikl přímo pro tuto práci a vznikl jako úplně nový projekt, kdy nebyly využity ani šablony, ani existující

přehrávače. Přehrávač je ale možné využít v jakémkoliv projektu a je schopný přehrávat i Smooth Streaming obsah. Tento přehrávač byl vytvořen v několika verzích tak, aby bylo možné testovat všechny výše zmíněné funkce. Byl vytvořen přehrávač se zapnutou podporou GPU i s vypnutou podporou GPU, s možností spuštění Out of Browser i bez ní, přehrávač vytvořený v Silverlightu 3 i v Silverlightu 4 Beta. Vždy se jedná o stejný přehrávač se stejnými funkcemi, ale upravený pro konkrétní účel. Obrázek z tohoto přehrávače viz níže Obrázek 5.

Webová stránka, která tuto Silverlight aplikaci obsahuje, neobsahuje nic jiného a nastavuje Silverlight pluginu, že má zobrazovat tzv. Framerate Counter (51), který zobrazuje počet snímků za vteřinu (nejedná se o snímky videa, ale o snímky celé Silverlight aplikace), použitou videopaměť v kilobajtech a další údaje. Pro zobrazení tohoto počítadla snímků musí být také u Silverlight pluginu aktivována podpora GPU (52), to se ale týká pouze objektu ve webové stránce, přehrávač toho vůbec využívat nemusí a tak je možné tohoto Framerate Counteru využít i při měření přehrávání s vypnutou podporou GPU. Tento Framerate Counter je vyznačen na obrázku níže i se zobrazenými údaji, které budou následně použity. Framerate Counter není možné spustit při běhu mimo prohlížeč (Out of Browser).



Obrázek 5: Použitý přehrávač a Framerate Counter

Kromě již zmíněného Framerate Counteru se jedná o program Správce úloh, který je v systému Windows integrován. Takže k měření nebudou využity žádné speciální nástroje, pouze nástroje, které poskytuje Silverlight a operační systém Windows.

5.2.3 Měřené atributy

V předchozím odstavci a na obrázku výše byly zmíněny atributy, které budou v experimentu měřeny. Těmito atributy jsou:

- Snímky za vteřinu, měřené pro Silverlight aplikaci – Framerate Counter
- Využití procesoru aplikací, která zobrazuje Silverlight aplikaci – Správce úloh systému Windows

V seznamu jsou uvedeny i nástroje, jimiž budou atributy měřeny, tyto nástroje byly popsány výše.

Tyto atributy dostačují k tomu, aby mohlo být posouzeno fungování Silverlight aplikace a její zatěžování systému. Při měření funkce při běhu mimo prohlížeč (Out of Browser) bude měřen pouze využití procesoru, protože v tomto módu není možné zapnout Framerate Counter.

5.2.4 Měření

Snímky za vteřinu a také využití procesoru jsou velice proměnlivé hodnoty, které se při přehrávání rychle mění. Při měření využití procesoru budou při měření zaznamenávány nejnižší a nejvyšší hodnoty, které budou naměřeny při přehrávání prvních 50 vteřin. Z těchto minimálních a maximálních hodnot budou vytvořeny a používány průměrné hodnoty.

Počet snímků za vteřinu se mění pomaleji a tento atribut není tak důležitý a tak je možné určit hodnotu, která odpovídá průměru.

Využitá video paměť se během přehrávání nemění a tak je měření této hodnoty snadné.

5.2.5 Testovací sestava

Procesor:	AMD Athlon 64 X2 – 2500 MHz
Paměť:	2048MB DDR II 667MHz
Grafická karta:	Integrovaná ATI Radeon X1250
	NVIDIA GeForce 9800 GTX+
Operační systém:	Windows 7 Professional 32-bit

Tabulka 1: Testovací sestava

Počítač, na kterém bylo měření prováděno, se po dobu přehrávání neměnil a nebyly spouštěny žádné služby, které by mohly nějakým způsobem ovlivnit naměřené hodnoty. Z tabulky výše je patrné, že měření bylo prováděno na dvou grafických kartách. Využití různých grafických karet bylo zvoleno proto, aby bylo možné sledovat závislost změn (s GPU a bez GPU) na výkonu grafické karty.

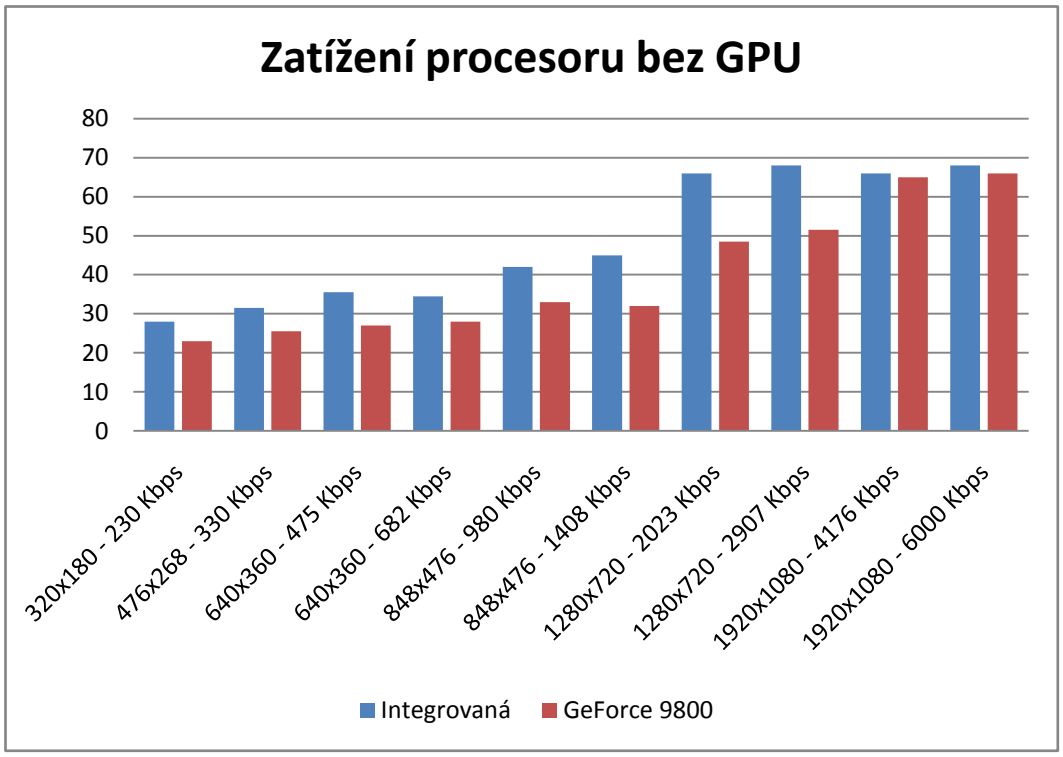
Grafická karta GeForce 9800 GTX+ mohla být připojena pouze na poloviční napájení, takže výkon této grafické karty nebyl maximální. Přesto byl podstatně vyšší než u grafické karty integrované na základní desce - ATI Radeon X1250.

5.3 Výsledky experimentu

5.3.1 Podpora HD videa

První měřenou možností Silverlightu je slibovaná podpora High Definition (HD) videa. Označení High Definition video označuje takové video, které má vysoké rozlišení, konkrétně se jedná o rozlišení 1280x720 (720p) a 1920x1080 (1080p) (63). V testovacích videích byla dvě videa s rozlišením 1280x720 (2023 a 2907 Kbps) a dvě videa s rozlišením 1920x1080 (4176 a 6000 Kbps).

Měření této možnosti bylo prováděno na dvou grafických kartách, aby bylo možné porovnat výsledky v závislosti na výkonnosti použitého hardwaru a bylo možné odpovědět na otázku, jak moc je možnost přehrávat HD video použitelná i na starších počítačích.



Obrázek 6: Podpora HD videa - zatížení CPU bez zapnuté podpory GPU

Z grafu vyplývá, že přehrávání HD videa v Silverlightu je možné. To je ale očekávaný výsledek. Další věcí, která je na grafu vidět je, že přehrávání HD videa podstatně více zatěžuje počítač uživatele a to jak na slabší grafické kartě, tak na silnější grafické kartě. Na silnější grafické kartě rozdíl není tak podstatný, ale přesto zde dochází ke skoku. Navíc je vidět, že zatížení počítače je více závislé na rozlišení přehrávaného videa než na datovém toku videa.

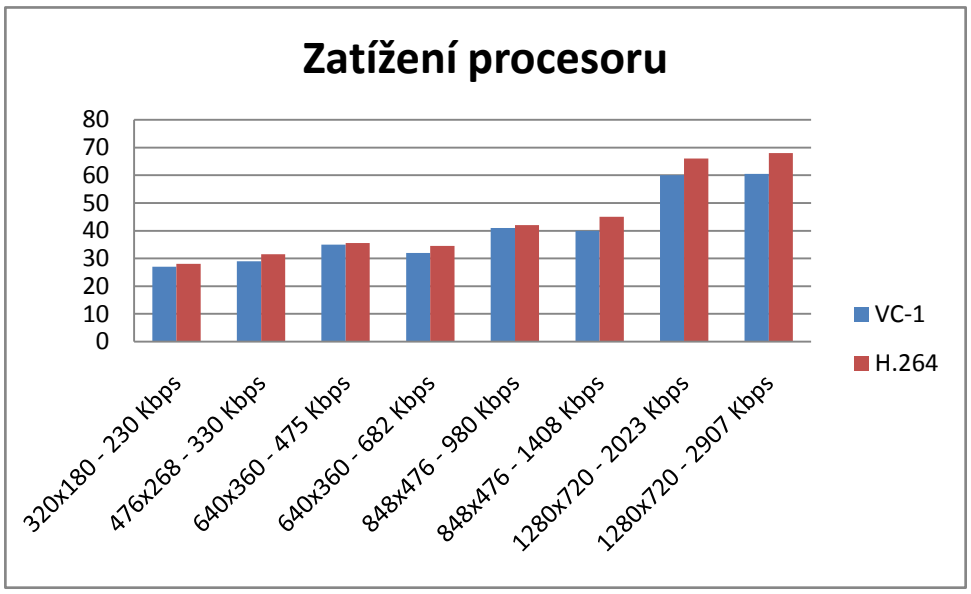
Je vidět, že zatížení procesoru při přehrávání HD videa (hlavně na slabší grafické kartě) je vysoké. To značně omezuje přehrávání, pokud by na počítači byly spuštěny další náročné úlohy. Také je vidět, že na slabší grafické kartě je rozdíl mezi zatížením počítače při přehrávání videa s rozlišením 1280x720 a 1920x1080 minimální. To vyplývá z toho, že přehrávání videa s rozlišením 1920x1080 není plynulé a dochází ke snižování zatížení procesoru.

Z naměřených (a v grafu zobrazených) výsledků vyplývá, že přehrávání HD videa v Silverlightu je možné, ale vyžaduje dostatečný výkon počítače, protože přehrávání tohoto videa je náročné. Silverlight v tomto konkrétním případě opravdu nabízí to, co slibuje.

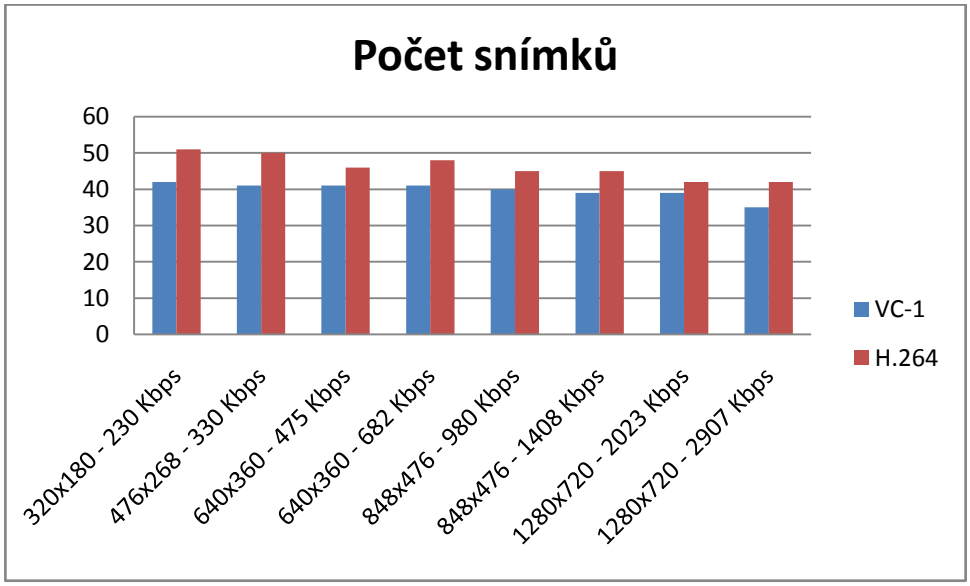
5.3.2 Podpora kodeků H.264 a VC-1

V tomto případě bylo ověřováno, zda Silverlight skutečně dokáže přehrávat oba moderní kodeky. Další věcí, která v souvislosti s těmito kodeky byla zkoumána, bylo porovnání těchto kodeků při přehrávání v Silverlightu.

V tomto případě nebudou zkoumána videa s rozlišením 1920x1080, protože jak bylo vidět v předchozím případě, u tohoto videa na integrované grafické kartě docházelo k neplynulému přehrávání a hodnoty by nebyly relevantní.



Obrázek 7: H.264 a VC-1 – Zatížení procesoru



Obrázek 8: H.264 a VC-1 – Počet snímků aplikace za vteřinu

V tomto případě je kromě zatížení počítače důležité i to, jaký výkon poskytuje Silverlight aplikace při přehrávání videa v těchto formátech. Tento výkon je reprezentován počtem snímků za vteřinu Silverlight aplikace. V tomto případě bylo měření prováděno pouze na jedné grafické kartě.

Opět je na grafech vidět, že Silverlight umožňuje přehrávání videí komprimovaných kodekem H.264 i kodekem VC-1. Takže se jedná o očekávaný výsledek.

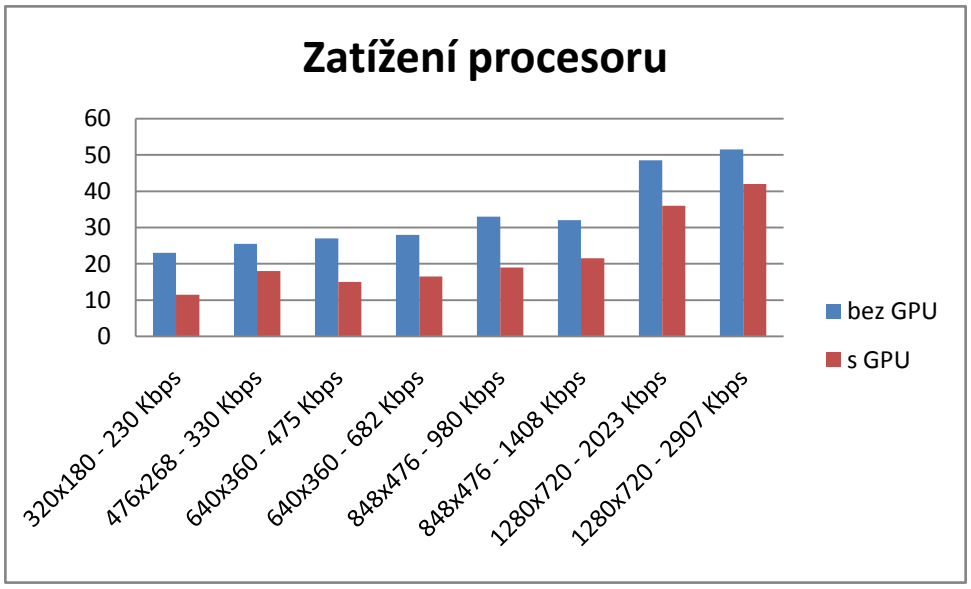
Další věcí je porovnávání obou kodeků. Z grafů vyplývá, že na všech osmi (pro každý kodek osm) testovacích videích bylo zatížení procesoru vyšší při přehrávání videa ve formátu H.264. Ovšem na druhou stranu přehrávání videí komprimovaných tímto kodekem poskytuje Silverlight aplikaci vyšší výkon, opět ve všech osmi případech. Jedná se o zajímavý výsledek, kdy z měření vyplývá, že přehrávání videí ve formátu VC-1 je na jednu stranu šetrnější k zatížení počítače, ale na druhou stranu neposkytuje tak vysoký výkon. Rozdíly jsou ovšem minimální.

Tento výsledek nemusí mít žádnou souvislost právě se Silverlightem, je zcela zřejmé z minimálních rozdílů, že Silverlight je stejně optimalizován pro jeden, i pro druhý kodek. Minimální rozdíly jsou s největší pravděpodobností způsobeny rozdílnou implementací obou kodeků.

5.3.3 Podpora GPU

V současné době je podpora GPU akcelerace po softwaru často vyžadována a je považována za velkou výhodu. Důležitost podpory GPU roste společně s rostoucím výkonem grafických karet. Ten je stále ještě využíván hlavně u počítačových her a u grafických aplikací. Z toho plyne, že po většinu času není procesor grafické karty (GPU) vytěžován tak, jak by být mohl. Ovšem aplikace, které tento procesor dovedou používat, získávají z počítače vyšší výkon, než kdyby využívaly pouze CPU. Aplikace mohou nabídnout výsledky dříve, mohou běžet plynuleji a navíc využívají celý potenciál, který počítač nabízí.

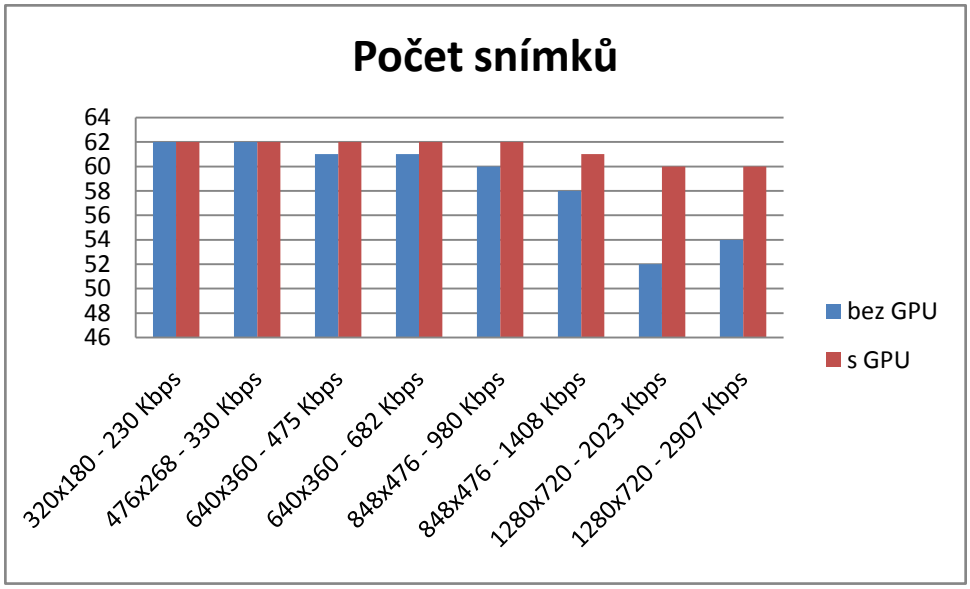
Silverlight podporu GPU slibuje a cílem tohoto měření bylo ověřit, zda při zapnuté GPU akceleraci je opravdu sníženo zatížení CPU a zda výkon Silverlight aplikace je v tomto případě vyšší. Tento vlastnost byla měřen na silnější grafické kartě, kde GPU je výkonnější a rozdíl by měl být více patrný. Opět jako u předchozího měření nebude při vyhodnocení uvažováno rozlišení 1920x1080, ze stejného důvodu.



Obrázek 9: Podpora GPU – zatížení procesoru

Na tomto grafu vidíme, že zatížení procesoru při zapnuté podpoře GPU je mnohem nižší než při vypnuté podpoře GPU. Tyto rozdíly jsou opravdu znatelné a GPU akcelerace má na zatížení procesoru velký vliv. Konkrétně se zatížení procesoru snížilo při datovém toku 230 Kbps o 50%, při 475 Kbps o 44% a při

rozlišení 1280x720 a datovém toku 2907 Kbps o 18%, což při vysokém zatížení procesoru rozhodně není zanedbatelné číslo! Z grafu jasně vyplývá, že pokud programátor má možnost v Silverlightu využít podporu GPU, měl by jí u aplikací přehrávajících video zapnout.



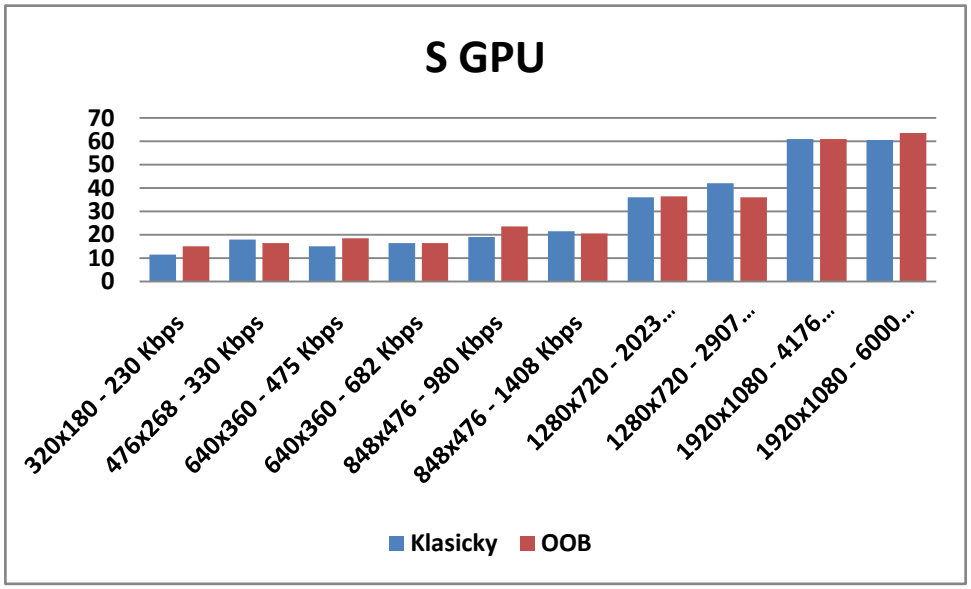
Obrázek 10: Podpora GPU – počet snímků

Výsledky tohoto grafu ukazují zvýšení výkonu Silverlight aplikace při zapnuté podpoře GPU. Jak je vidět, rozdíly mezi naměřenými hodnotami při zapnuté a vypnuté podpoře GPU nejsou nijak propastné, hlavně u videí s nižším rozlišením. Z toho by se mohlo zdát, že sice při zapnuté podpoře GPU je sníženo celkové zatížení počítače, ale výkon se prakticky nezvedne. Ale v tomto případě je potřeba se na graf podívat blíže. Jak je vidět, maximální dosažená hodnota je 62 snímků za vteřinu a to i u těch nejnižších kvalit videa. Z toho vyplývá, že 62 je maximální hodnota, kterou měřená Silverlight aplikace dokáže. Více snímků ani zdaleka není potřeba. A při bližším prozkoumání grafu je zřejmé, že i přesto, že při vypnuté podpoře GPU počet snímků začíná klesat a při u HD videa už se pohybuje okolo 52 snímků, tak při zapnuté podpoře se stále drží u maxima. Z toho vyplývají nižší rozdíly. Pokud by byl počítač zatížen například spuštěním druhé Silverlight aplikace, pravděpodobně by se počet snímků snížil a rozdíly by byly patrnější. I přesto je z grafu zřejmé, že při zapnuté podpoře GPU aplikace podává téměř maximální výkon

u všech testovacích videí. U HD videa se jedná o zvýšení výkonu o 15% a to samozřejmě je nezanedbatelné číslo.

5.3.4 Out of Browser

Experiment zkoumající rozdíl mezi Silverlight aplikací běžící v prohlížeči (při měření Internet Explorer 8) a v módu Out of Browser, v aplikaci SLLauncher. U Out of Browser aplikací není možné využít nástroj Framerate Counter a tak nemohl být změřen výkon Silverlight aplikace a bylo změřeno pouze zatížení procesoru.



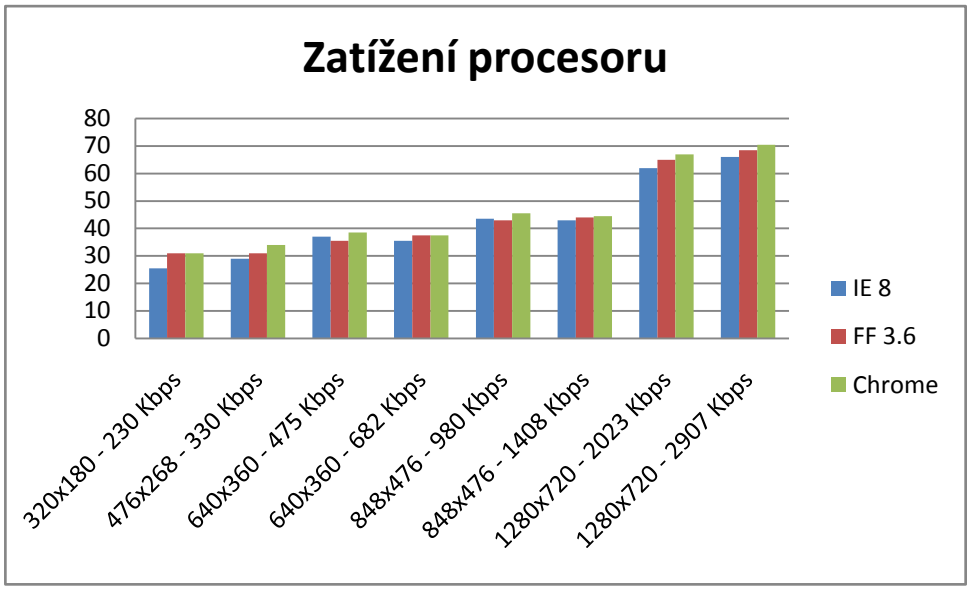
Obrázek 11: Out of Browser – zatížení procesoru

Z grafu je na první pohled zřejmé, že rozdíly mezi naměřenými hodnotami jsou minimální a navíc není možné říci, že jeden nebo druhý přístup více nebo méně zatěžuje systém. To proto, že zatížení je v některých případech vyšší při Out of Browser módu a někdy při klasickém běhu uvnitř prohlížeče.

5.3.5 Podpora různých prohlížečů

Silverlight oficiálně nabízí podporu různých prohlížečů (viz příslušná kapitola) a v tomto experimentu je ověřována podpora prohlížečů Internet Explorer 8, Firefox 3.6 a Google Chrome 4. Nejedná se pouze o ověřování toho, zda daný prohlížeč dokáže spouštět Silverlight aplikace, ale také jsou prohlížeče porovnávány z pohledu zatížení počítače a výkonu Silverlight aplikace. V tomto případě byla poprvé při

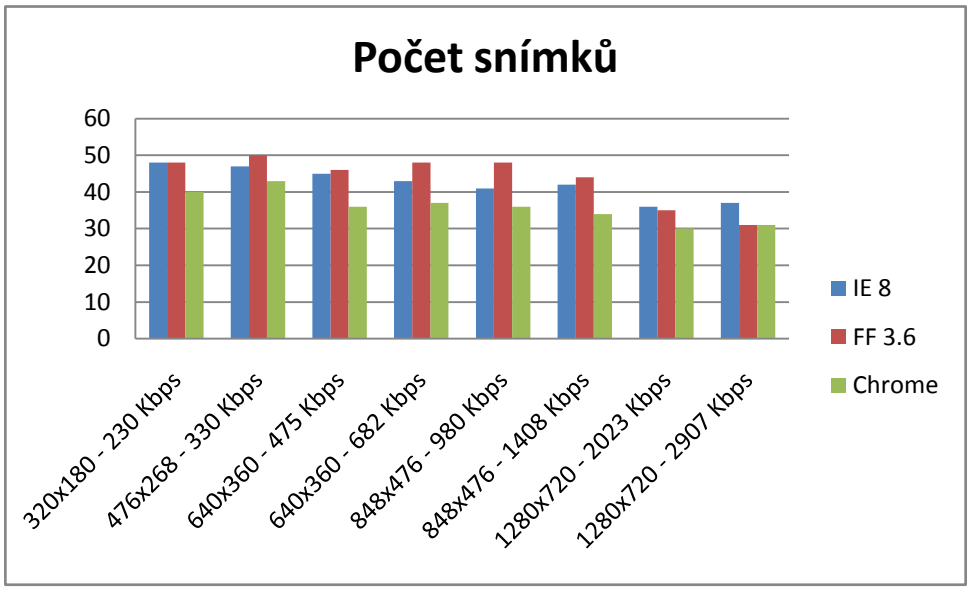
měření využívána aplikace vytvořená pomocí beta verze Silverlightu 4, který nabízí oficiální podporu pro Google Chrome 4. Stejně jako v předchozích případech nebyla grafu využívána data naměřená při přehrávání videa s rozlišením 1920x1080.



Obrázek 12: Různé prohlížeče – zatížení procesoru

Tento graf ukazuje, že zatížení procesoru při spuštění Silverlight aplikace v různých prohlížečích se mění jen velice málo. Rozdíly mezi jednotlivými prohlížeči jsou minimální a s největší pravděpodobností jsou způsobeny samotnými prohlížeči, které samozřejmě svým během zatěžují počítač každý jinak. Minimální rozdíly jsou způsobeny tím, že Silverlight plugin spouštěný v těchto prohlížečích je stále stejný.

Tyto výsledky by měly být podpořeny také grafem, který ukazuje výkon Silverlight aplikace v různých prohlížečích. Ovšem z tohoto grafu vyplývá, že tomu tak není, protože u prohlížeče Google Chrome je počet snímků ve všech případech nižší než u ostatních prohlížečů. Navíc tyto rozdíly nejsou minimální.



Obrázek 13: Různé prohlížeče – počet snímků

Tento nižší výkon při běhu v prohlížeči Google Chrome může být způsoben především tím, že Silverlight oficiálně podporuje tento prohlížeč až od verze 4, která byla v době testování dostupná pouze jako beta určená pro vývojáře. Proto nemusí být podpora pro tento prohlížeč optimalizována stejně jako u dlouho podporovaných prohlížečů Internet Explorer a Firefox.

6 Silverlight a práce se zařízeními na straně klienta

6.1 Co je považováno za zařízení na straně klienta?

Zařízení na straně klienta jsou obecně ta zařízení, která jsou připojena k počítači uživatele aplikace. V souvislosti s multimédii a webovými technologiemi jsou využívána hlavně dvě zařízení:

- Mikrofon
- Kamera

Mikrofony jsou samozřejmě využívány při práci se zvukem. Zvuk na straně klienta může být nahráván, přenášen k jinému klientovi (telefonní hovory apod.), upravován různými efekty. Možností při práci s mikrofonem je spousta, vždy záleží na konkrétních požadavcích a konkrétním použití.

U kamer se jedná o práci s videem. Toto video může být nějakým způsobem zpracováváno, může být přenášeno k jinému klientovi (video hovory, live streaming a další). Stejně jako u mikrofonů je způsobů použití mnoho, takže záleží na konkrétním scénáři.

Ve většině případů se na straně klienta pracuje s webkamerami, které mohou být, stejně jako mikrofony, integrované a externí. Integrované webkamery a mikrofony bývají většinou dostupné v noteboocích a někdy mohou být součástí monitorů. U stolních počítačů se většinou jedná o externí zařízení, která se připojují například přes USB zařízení.

6.1.1 Práce s webkamerami a mikrofony

Při práci s těmito zařízeními je potřeba stále počítat s jistými omezeními. Tato omezení jsou způsobena především dvěma důvody:

- Nikdy není jisté, že klient má tato zařízení k dispozici a jsou připojená
- Bezpečnost

První omezení plyne z toho, že ne každý má na svém počítači připojenou webkameru a ne každý má připojený mikrofon. Takže vývojář, který ve svém programu chce pracovat s těmito zařízeními, takže například chce uživateli

zobrazovat obraz z webové kamery, musí ošetřit také možnost, že zařízení není na straně klienta k dispozici. V tom případě je nutné uživatele informovat, že zařízení není k dispozici a pro správné fungování programu musí toto zařízení dokoupit nebo doinstalovat. Rovněž je nutné zajistit, aby při pokusu pracovat se zařízením, které není k dispozici, nedošlo k nezachycené výjimce a tím k pádu aplikace.

Druhým omezením je bezpečnost, protože webové aplikace, což Silverlight aplikace bezpochyby jsou, běží v bezpečnostním sandboxu prohlížeče (jak už bylo řečeno v kapitole o Silverlightu). Bezpečnost je samozřejmě důležitá, protože webové aplikace běžící v prohlížeči nemohou s počítačem klienta manipulovat zcela podle uvážení programátora. To je nutné protože ne každý programátor má čisté úmysly a mohl by tímto způsobem uživatele ohrozit, přistupovat k citlivým datům, získat počítač pod kontrolu atd. Běh v bezpečnostním sandboxu prohlížeče zajišťuje, že manipulace programu s počítačem bude omezena. Přístup k zařízením, která jsou k počítači uživatele připojena, vyžaduje určitá další oprávnění. Proto je nutná interakce s uživatelem, který přístup k těmto zařízením musí povolit. Při vytváření aplikací, které s mikrofony a webkamerami pracují, je nutné ošetřit také možnost, kdy uživatel přístup nepovolí (podobně jako v případě, kdy zařízení není k dispozici). Tento popsaný přístup se týká Silverlightu, ale podobně pracují i ostatní technologie, například Flash.

Další věcí, se kterou je v takovýchto aplikacích potřeba pracovat, je možnost připojení více než jednoho mikrofonu nebo webkamery. Tento stav může nastat například, když má uživatel integrovanou webkameru v notebooku a zároveň má připojenu i externí webkameru. V takové aplikaci je užitečné dát uživateli možnost vybrat zařízení, které se bude využívat.

6.1.2 Použití zařízení na straně klienta

Jak už bylo řečeno u popisu kamery a mikrofonu, tak jejich použití je velice široké a je závislé na požadavcích na aplikaci. Ovšem je možné vyjmenovat několik scénářů, kde je přístup ke kamerám a mikrofonům často využíván:

- Komunikace
- Hry
- Efekty

Komunikace, tedy první bod patří mezi nejčastější scénáře, při kterých se využívají mikrofony a také webkamery. Mikrofony se dají využít samostatně, když jsou využity pro telefonní aplikace. Slouží ke komunikaci mezi uživateli, kteří si přes konkrétní software mohou telefonovat. Podobně, při použití webkamery se mohou uživatelé i vidět a jedná se o video hovory, kdy se mezi uživateli nepřenáší pouze zvuk, ale i video. To umožňuje lepší a realističtější spojení mezi uživateli. Pomocí webkamery se dá posílat také samotný obraz, který může doplnit obyčejný komunikační program, kdy si uživatelé posílají text.

Webkamera se dá využít i při hrách, kdy obraz z webkamery nemusí být nikam odesílán. V takovýchto případech mohou být například porovnávány jednotlivé snímky a tak může být rozeznáván pohyb nebo zobrazení určitých vzorů. Tímto způsobem je možné vytvořit například hry, které se ovládají pohyby rukou i přesto, že nevyužívají dotykového displeje. Další možností je obohacení her o obraz z webkamery, kdy z tohoto obrazu může být vytvořeno například puzzle a to se dá skládat. Možností, jak obohatit hry o obraz z kamery je velké množství a záleží pouze na fantazii a schopnostech autorů.

Poslední bod, efekty, úzce souvisí s využíváním webkamer při hraní. Jedná se o to, že se dají zachytávat jednotlivé snímky z kamery, nebo sekvence snímků. Takto zachycené snímky mohou být pomocí různých efektů upravovány, ukládány atd. Částečně to souvisí s hraním, kdy tyto aplikace nefungují jako profesionální nástroje, ale jsou to aplikace určené pro zábavu a volný čas.

6.2 Silverlight a přístup k zařízením na straně klienta

6.2.1 Silverlight 3

V Silverlightu verze 3, aktuální plné verzi (verze 4 je v současné době dostupná jako Release Candidat nebo Beta verze, zatím je ve verzi určené především pro vývojáře), není možné k zařízením na straně klienta přistupovat žádným způsobem.

6.2.2 Silverlight 3 a DirectX

Pro splnění cílů vytyčených pro tuto práci byly uvažovány další možnosti, jak využít aktuální verzi Silverlightu (verzi 3), pro přístup k zařízením na straně klienta. Hlavní možností, která byla v této souvislosti uvažována, bylo využití knihoven DirectX k přístupu k hardware zařízením a práci s těmito zařízeními. Jednalo by se o elegantní způsob, který je využíván například v souvislosti s technologií Windows Presentation Foundation (60, 61).

Bohužel tato možnost, která je použita ve WPF aplikacích, není využitelná v Silverlight aplikacích. Je to z toho důvodu, že knihovny DirectX jsou určeny pro operační systém Windows a Silverlight je určen nejen pro Windows a tak Silverlight nepodporuje práci s DirectX. V Silverlight aplikacích není možné využívat stejné knihovny tříd, které je možné využívat ve Windows aplikacích. Tento rozdíl je dán omezeními webových aplikací, která byla popsána výše.

Takže ani tato možnost nemohla být využita při práci Silverlightu a zařízeními na straně klienta.

6.2.3 Silverlight 4

Na konferenci Microsoftu PDC 2009 (Professional Developers Conference) v Los Angeles byla dne 18. listopadu 2009 vydána Beta verze Silverlightu 4. Tato verze byla určena především pro vývojáře, aby se seznámili s novými možnostmi, které Silverlight 4 nabízí (62). Na konferenci MIX 2010, 15. března 2010 byla vydána verze Silverlight 4 RC, která je v současnosti nejaktuálnější verzí Silverlightu.

Jednou z možností, které Silverlight 4 nabízí je přístup k zařízením na straně klienta – k mikrofonu a kameře. Neschopnost přistupovat k těmto zařízením byla

předchozím verzím Silverlightu vyčítána v různých diskuzích a tak se Microsoft rozhodl implementovat i tuto možnost. Možnost práce s těmito zařízeními přímo v Silverlightu 4 dovoluje splnit cíle učené pro tuto práci.

V Silverlightu 4 (dále pouze Silverlight) je možné k těmto zařízením přistupovat několika způsoby, které jsou popsány dále.

6.3 Práce se zařízeními na straně klienta v Silverlightu

V předchozí kapitole bylo zmíněno, že pracovat v Silverlightu s hardwarovými zařízeními jako jsou mikrofony a kamery, lze pouze za použití Silverlightu ve verzi 4. Dále bylo řečeno, že je možné pracovat s těmito zařízeními několika způsoby. Tyto způsoby budou popsány v této kapitole a týkají se těchto tříd:

- **CaptureSource** – reprezentuje zachytávací (audio/video) zařízení a umožňuje:
  - zachytávat snímky z kamery jako instanci třídy WriteableBitmap
  - nastavit instanci třídy CaptureSource jako zdroj pro instanci třídy VideoBrush
  - nastavit instanci třídy CaptureSource jako zařízení použité v instanci třídy AudioSink/VideoSink
- **VideoBrush** – spolu s využitím třídy CaptureSource umožňuje obraz zachytávaný webkamerou použít jako štětec, kterým se „natře“ nějaký element, typicky se jedná o objekty typu Rectangle, TextBlock atd.
- **AudioSink** – abstraktní třída, která slouží pro práci se syrovými (RAW) audio daty
- **VideoSink** – abstraktní třída, která slouží pro práci se syrovým (RAW) video daty

Toto jsou nejdůležitější třídy při práci se zařízeními na straně klienta. Důležité je také umět používat abstraktní třídu MediaStreamSource, která umožňuje zpracovávat syrová RAW data, která se díky MediaStreamSource dají v Silverlightu přehrávat pomocí objektu MediaElement. Ke třídě MediaStreamSource a k přehrávání videa pomocí objektu MediaElement viz první část této práce.

6.3.1 Třída CaptureSource

Třída, která poskytuje metody, které pracují s audio a video zařízeními přiřazenými do instance této třídy. Instancím této třídy se dá přiřadit používané audio zařízení do vlastnosti AudioCaptureDevice. Tato vlastnost určuje zařízení, které bude aplikace využívat pro práci se zvukem. Podobným způsobem se nastavuje používané video zařízení, které se přiřazuje vlastnosti VideoCaptureDevice.

Samotná třída CaptureSource poskytuje tři metody pro práci se zařízeními připojenými pomocí vlastností VideoCaptureDevice a AudioCaptureDevice. Těmito metodami jsou:

- **Start()** – metoda zahájí snímání ze všech snímacích zařízení, které jsou přiřazené k této instanci třídy CaptureSource, po provedení této metody může být instance třídy CaptureSource nastavena jako zdroj instanci třídy VideoBrush, také může být využita metoda CaptureImageAsync()
- **Stop()** – metoda zastaví snímání všech připojených zařízení
- **CaptureImageAsync()** – metoda, která inicializuje asynchronní požadavek na získání obrázku z webkamery, obrázek (WriteableBitmap) vrácený jako odpověď je vrácen v události CaptureImageCompled jako CaptureImageCompletedEventArgs.Result, pomocí této metody je možné získat aktuální obrázek snímaný webkamerou

Tato třída poskytuje první možnost přístupu k zařízením na straně klienta. Touto první možností je zachytávat snímky z webkamery pomocí metody CaptureImageAsync, která požaduje aktuální snímek z webkamery. Aktuální snímek z webkamery je vrácen tak, že je vyvolána událost CaptureImageCompled, které se obrázek (instance třídy WriteableBitmap) předává jako argument. S těmito obrázky je možné pracovat jako s jinými obrázky typu WriteableBitmap, lze je zobrazovat a odesílat atd. Jednou možností jak pomocí této metody zobrazovat video z webkamery je využít časovač, který bude v pravidelném intervalu tyto snímky zachytávat a zobrazovat, takže pro uživatele zobrazování snímků bude působit jako přehrávání videa. Samozřejmě je možné zachytávat snímky jen na požádání a určitým způsobem je zpracovávat, například na ně aplikovat efekty.

6.3.2 Třída VideoBrush

Druhou popisovanou možností, jak pracovat s webkamerou, je využití třídy VideoBrush. Tato třída dědí ze třídy TileBrush, která dědí ze třídy Brush, to umožňuje využívat instance třídy VideoBrush stejně, jako jakýkoliv jiný Brush. Klasicky používanými třídami zděděnými ze třídy Brush jsou SolidColorBrush, LinearGradientBrush a existují i další, které je možné využít. Instance tříd zděděných ze třídy Brush se využívají pro nastavení štětce, kterým je možné „natřít“ pozadí, popředí, rámeček, výplň atd. Záleží na konkrétně používaném elementu a jeho vlastnostech.

S instancemi třídy VideoBrush se dá pracovat dvěma způsoby, jedním způsobem je použít jako zdroj instanci třídy MediaElement a druhým způsobem je použít instanci třídy CaptureDevice.

Pokud je v těchto kapitolách popisována práce s webkamerami a mikrofony, je první způsob (jako zdroj nastavený MediaElement) nepodstatný. Důležitý je druhý způsob, kdy se instanci třídy VideoBrush nastavuje pomocí metody SetSource(CaptureDevice) zařízení, které poslouží jako zdroj videa.

Nejjednodušším způsobem jak přistupovat k webkameře a zobrazovat snímané video je popisovaným způsobem vytvořit VideoBrush a nastavit mu jako zdroj vytvořený zdroj (CaptureSource). Touto instancí třídy VideoBrush lze „natřít“ například výplň obdélníku (Rectangle.Fill), poté tento obdélník bude zobrazovat snímané video. Stejným způsobem je možné použít jako „plátno“ například elipsu (Ellipse). Ale není nutné video z webkamery používat jen takto, další možností je takto „natřít“ i text a další vlastnosti.

Tento přístup je velice jednoduchý a jedná se o nejsnazší možnost jak uživateli zobrazit na straně uživatele video snímané webkamerou. Problémem je, že video je zobrazeno pouze uživateli, u kterého je aplikace spuštěna. Opět je možné s tímto obrazem pracovat pomocí efektů, animací atd., ale není možné ho nijak zaznamenávat (stejně jako u metody CaptureSource.CaptureImageAsync()).

Tento přístup je vhodný především pro hry a další aplikace určené pro zábavu. Není možné jej použít pro komunikační scénáře, protože s videem se pracuje pouze jako se štětcem a ne jako s videem nebo daty.

Ukázka použití třídy VideoBrush:

```
            CaptureSource source = new CaptureSource()
            {
                VideoCaptureDevice =
CaptureDeviceConfiguration.GetDefaultVideoCaptureDevice()
            };

            VideoBrush brush = new VideoBrush();
            brush.SetSource(source);
            Rectangle.Fill = brush;
            Ellipse.Fill = brush;
            Text.Foreground = brush;

            source.Start();
```

6.3.3 Třída AudioSink

Tato abstraktní třída poskytuje přístup k audio zařízení. Třídy zděděné od této třídy umožňují Silverlight aplikacím přistupovat přímo k datům získaným z audio zařízení. Jedná se o syrová (RAW) data, se kterými je možné pracovat. Třídám zděděným ze třídy AudioSink je nutné nastavit zařízení, které bude použito ke snímání zvuku. Nastavení tohoto zařízení se provádí tak, že se vlastnosti CaptureSource nastaví instance třídy CaptureSource (viz výše). Při práci s touto třídou je potřeba implementovat abstraktní metody, které určují, jak bude AudioSink pracovat se syrovými audio daty. Těmito abstraktními metodami jsou:

- OnCaptureStarted()
- OnCaptureStopped()
- OnFormatChange(AudioFormat)
- OnSamples(long, long, byte[])

Metoda OnCaptureStarted()

Tato metoda je volána, když audio zařízení začne snímat zvuk. V této metodě je potřeba vykonat všechny operace, jež musí být vykonány při spuštění. Samozřejmě prováděné operace závisí na použití aplikace, takže pro tuto kapitolu bude uvažován příklad se zápisem audio dat do paměti (MemoryStream), jedná se o nejjednodušší

způsob práce AudioSinku a tento příklad bude fungovat, pokud zapsaná data budou převedeny na formát, který přehrávače umí přehrát. V této metodě bychom mohli chtít inicializovat proměnné využívané v dalších metodách, konkrétně se jedná o MemoryStream, do kterého se budou zapisovat audio data. V jiných příkladech bychom mohli chtít zapsat nějakou počáteční značku nebo třeba hlavičku s metadaty (například autor, datum vytvoření atd., závisí na formátu zpracovávaných dat).

Příklad:

```
protected override void OnCaptureStarted()
{
    stream = new MemoryStream();
}
```

Metoda OnCaptureStopped()

Tato metoda je volána, když audio zařízení ukončí snímání zvuku. V této metodě je potřeba provést všechny operace, které jsou nutné k ukončení zaznamenávání zvuku nebo k uzavření již zapsaných dat atd. Prováděné operace v této metodě samozřejmě opět závisí na konkrétním scénáři použití. Z tohoto důvodu jsou tyto metody abstraktní, protože každý scénář použití této třídy je jiný a implementace tříd zděděných ze třídy AudioSink je vždy závislá na konkrétním scénáři. Proto i jednotlivé metody jsou závislé na scénářích. Jedním z příkladů by mohlo být zapsání uzavírací značky při zápisu v nějakém speciálním formátu. V konkrétním případě pro tuto kapitolu se pouze ukončí stream s audio daty.

Příklad:

```
protected override void OnCaptureStopped()
{
    stream.Close();
}
```

Metoda OnFormatChange(AudioFormat)

Tato metoda je volána, když audio zařízení informuje o změně formátu snímaného audia. Parametr předávaný této metodě je typu AudioFormat a popisuje formát dat (například počet kanálů zvuku), která jsou zařízením snímána. V této metodě je potřeba vykonat operace, které nastaví parametry závislé na použitém formátu

zvuku. V příkladu pro tuto kapitolu není potřeba provést žádné operace a tak se jen aktuální audio formát uloží do proměnné.

Příklad:

```
protected override void OnFormatChange(AudioFormat audioFormat)
{
    format = audioFormat;
}
```

Metoda OnSamples(long, long, byte[])

Metoda je volána, když audio zařízení dokončí audio vzorek. Parametry předávané této metodě určují čas pořízení tohoto vzorku, dobu trvání vzorku a samotná data vzorku předávaná jako pole bajtů. Této metodě je předáván aktuální kompletní audio vzorek s informacemi, které aktuální vzorek popisují. Tato metoda může být považována za nejdůležitější proto, že je nejčastěji volána a vykonává operace zpracovávající vzorky. Vzorky mohou být komprimovány, přesouvány, odesílány na server nebo zapisovány. Zapisování dat vzorku je to, co provádí příklad pro tuto kapitolu, který pole bajtů vezme a zapíše ho do streamu. Tento stream se skládá z jednotlivých vzorků zapsaných přímo za sebou tak, že tvoří stream reprezentující celé zaznamenané audio.

Příklad:

```
protected override void OnSamples(long sampleTimeInHundredNanoseconds,
    long sampleDurationInHundredNanoseconds, byte[] sampleData)
{
    stream.Write(sampleData, 0, sampleData.Length);
}
```

6.3.4 Třída VideoSink

Abstraktní třída VideoSink je velice podobná abstraktní třídě AudioSink. Třídy odvozené od této abstraktní třídy se používají pro přístup k video zařízením, typicky se jedná o webkamery. Instancím těchto tříd je nutné přiřadit zařízení, se kterým se bude pracovat. Toto nastavení se provádí ekvivalentně třídám odvozeným od AudioSink. Stejně jako třída AudioSink, poskytuje i tato třída přístup k syrovým RAW datům, to znamená datům nijak nekomprimovaným. Protože rozdíly mezi

třídou VideoSink a AudioSink jsou opravdu minimální, tak popis této třídy bude o něco stručnější. Opět je potřeba implementovat abstraktní metody:

- OnCaptureStarted()
- OnCaptureStopped()
- OnFormatChange(VideoFormat)
- OnSample(long, long, byte[])

Jak je vidět, abstraktní metody jsou stejné jako u třídy AudioSink a jediným viditelným rozdílem je, že metodě OnFormatChange je jako parametr předávána instance třídy VideoFormat (u AudioSink je to instance třídy AudioFormat). Tyto metody jsou volány ve stejných situacích jako u třídy AudioSink a tak není potřeba opakovat stejný popis. Popis těchto metod viz předchozí kapitola – AudioSink. Stejně tak u ukázkového příkladu je možné využít podobný scénář jako ve zmiňované kapitole – zápis syrových RAW dat do streamu v paměti (MemoryStream). Protože popis metod by se opakoval, tak je jako příklad přiložen celý zdrojový kód třídy zděděné z abstraktní třídy VideoSink.

Příklad:

```
public class MyVideoSink : VideoSink
{
    MemoryStream stream;
    VideoFormat format;

    protected override void OnCaptureStarted()
    {
        stream = new MemoryStream();
    }

    protected override void OnCaptureStopped()
    {
        stream.Close();
    }

    protected override void OnFormatChange(VideoFormat videoFormat)
    {
        format = videoFormat;
    }
    protected override void OnSamples(long sampleTimeInHundredNanoseconds,
        long sampleDurationInHundredNanoseconds, byte[] sampleData)
    {
        stream.Write(sampleData, 0, sampleData.Length);
    }
}
```

6.3.5 Povolení přístupu k zařízením na straně klienta

Jak už bylo řečeno v kapitole obecně popisující práci se zařízeními na straně klienta, tak pokud jsou tato zařízení využívána ve webových aplikacích (jako jsou Silverlight aplikace), musí aplikace získat od uživatele povolení přistupovat k těmto zařízením. Důvod byl také popsán výše – bezpečnost. Bezpečné aplikace k těmto zařízením nepřistupují zcela bez vědomí uživatele. Aplikace vytvořené pomocí technologie Silverlight z tohoto pohledu jsou bezpečné (bezpečnost Silverlight aplikací není tématem této práce), protože v Silverlight aplikacích není možné bez povolení uživatele k webkamerám a mikrofonům přistupovat. Uživatel může přístup odmítnout, pokud nesouhlasí nebo nevyvolal tuto akci úmyslně.

V Silverlightu existují dvě možnosti, jak uživatel povolí přístup k zařízením na straně klienta:

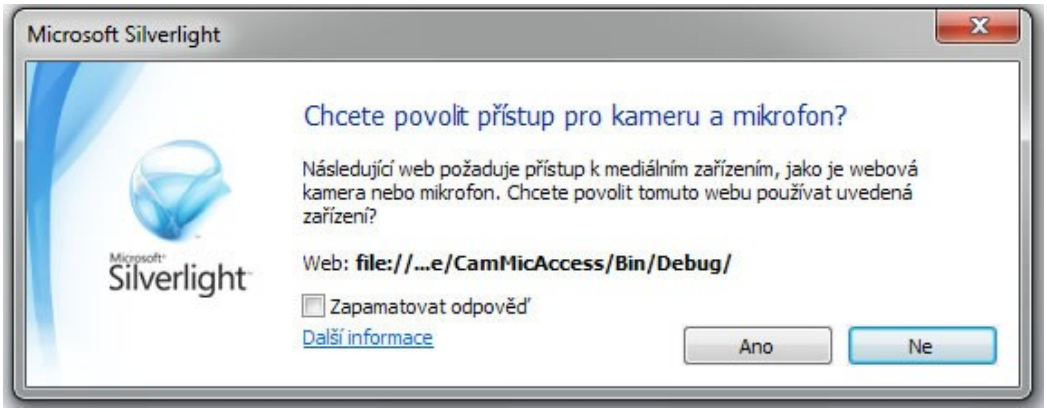
- Vyvolání požadavku a povolení či nepovolení pomocí dialogového okna „Chcete povolit přístup pro kameru a mikrofon?“ pomocí třídy CaptureDeviceConfiguration
- Spouštění Silverlight aplikace v módu Out of Browser (aplikace instalovaná na počítači uživatele) se zvýšenou důvěrou (elevated trust mód)

Povolení pomocí třídy CaptureDeviceConfiguration

Jedná se o základní možnost, jak povolit přístup k zařízením na straně klienta v Silverlight aplikaci. Silverlight poskytuje možnost zjistit nastavení snímacích zařízení, mezi tato nastavení patří i informace o tom, zda byl povolen přístup ke kamerám a mikrofonům. Pro přístup k těmto nastavením se využívá pomocná statická třída CaptureDeviceConfiguration. Tato třída poskytuje pět veřejných metod a jednu veřejnou vlastnost jen pro čtení, která obsahuje informaci o tom, jestli už uživatel povolil přístup k mikrofonům a kamerám na svém počítači. Pokud je tato vlastnost false, znamená to, že přístup zatím nebyl povolen nebo byl odepřen.

Požadavek o povolení se vyvolává pomocí zmíněné statické třídy CaptureDeviceConfiguration a její veřejné metody RequestDeviceAccess(), která vyvolá a zobrazí uživateli dialogové okno „Chcete povolit přístup pro kameru a

mikrofon?“. V tomto dialogovém okně uživatel může povolit nebo odepřít přístup k těmto zařízením, a pokud této aplikaci důvěřuje, může zaškrtnout zaškrtávací políčko pro zapamatování odpovědi. Toto zapamatování umožňuje, aby pro tuto aplikaci už nemusel povolovat nebo zakazovat přístup, ale přístup byl pro tuto aplikaci povolen vždy nebo nikdy. Toto nastavení se uloží do nástroje Konfigurace produktu Microsoft Silverlight (sekce Oprávněni). Tento nástroj je možné vyvolat přímo z operačního systému. Ve Windows stačí spustit soubor Silverlight.Configuration.exe. Druhou možností jak spustit tento konfigurační soubor je pravým tlačítkem kliknout na Silverlight aplikaci a kliknout na Silverlight (to je možné, jen pokud není programově zpracováváno kliknutí pravým tlačítkem myši).



Obrázek 14: Dialogové okno „Chcete povolit přístup pro kameru a mikrofon?“

Pokud je přístup k zařízením na straně klienta povolen, je s nimi možné pracovat výše popsanými způsoby.

Ukázka povolení přístupu k zařízením na straně klienta – pouze povolí přístup k zařízením, ale nijak s nimi nepracuje:

```
bool isAccess = CaptureDeviceConfiguration.AllowedDeviceAccess;

if (!isAccess)
{
    bool ok = CaptureDeviceConfiguration.RequestDeviceAccess();

    if (ok)
    {
        MessageBox.Show(
            "Přístup k mikrofonům a kamerám byl povolen.",
            "Povolení přístupu", MessageBoxButton.OK);
    }
    else
    {
        MessageBox.Show(
            "Přístup k mikrofonům a kamerám byl odepřen.",
            "Povolení přístupu", MessageBoxButton.OK);
    }
}
else
{
    MessageBox.Show(
       "Přístup k mikrofonům a kamerám byl již povolen.",
        "Povolení přístupu", MessageBoxButton.OK);
}
```

Spouštění v módu Out of Browser se zvýšenou důvěrou

Druhou možností, jak Silverlight aplikace mohou získat přístup k zařízením na straně klienta, je instalace aplikace do počítače uživatele. Jedná se o mód Out of Browser. Pokud vývojář Silverlight aplikace povolí její instalaci do počítače uživatele, ne jen prohlížení v prohlížeči, má možnost vyžadovat od uživatele zvýšenou důvěru. Tato zvýšená důvěra je poté uživatelem akceptována instalací aplikace.

Pokud uživatel vydavateli aplikace, a aplikaci, důvěřuje tak, že si tuto aplikaci se zvýšenou důvěrou nainstaluje do svého počítače, tak automaticky aplikaci povoluje přístup k mikrofonům a webkamerám.

Tato instalace se zvýšenou důvěrou má stejný význam jako předchozí způsob se zapamatováním volby. To znamená, že Silverlight aplikace může využívat webkamery a mikrofony bez toho, aby uživatel byl pokaždé žádán o povolení, povolením byla instalace aplikace.

6.3.6 Výběr používaného zařízení

Už výše bylo zmíněno, že uživatel může mít k počítači připojeno několik zařízení pro snímání videa i zvuku a je obvykle vhodné umožnit uživateli využívat zařízení, které si vybere. Silverlight toto umožňuje, samozřejmě vždy záleží na vývojáři aplikace, zda těchto možností využije a uživateli umožní výběr zařízení. Vždy je totiž k dispozici možnost využít defaultně vybrané zařízení a tak se vyhnout výběru používaného zařízení.

Silverlight v tomto případě umožňuje dvě možnosti, jak zařízení nastavit:

- Programově – přímo v Silverlight aplikaci
- Konfigurace Silverlightu - nástroj Konfigurace produktu Microsoft Silverlight

Přímo v Silverlight aplikaci

V Silverlightu je možné získat kolekci dostupných zařízení a to jak mikrofonů, tak webkamer. Opět se využívá pomocné statické třídy CaptureDeviceConfiguration, která k tomuto účelu poskytuje dvě veřejné metody – GetAvailableAudioCaptureDevices(), GetAvailableVideoCaptureDevices(). Tyto metody vrací kolekce dostupných zařízení. Zařízení jsou v kolekci reprezentována instancemi třídy AudioCaptureDevice pro audio zařízení a VideoCaptureDevice pro video zařízení. Kolekce jsou pouze pro čtení, protože obsahují dostupná zařízení připojená k počítači a není možné je programově přidávat, odebírat nebo měnit.

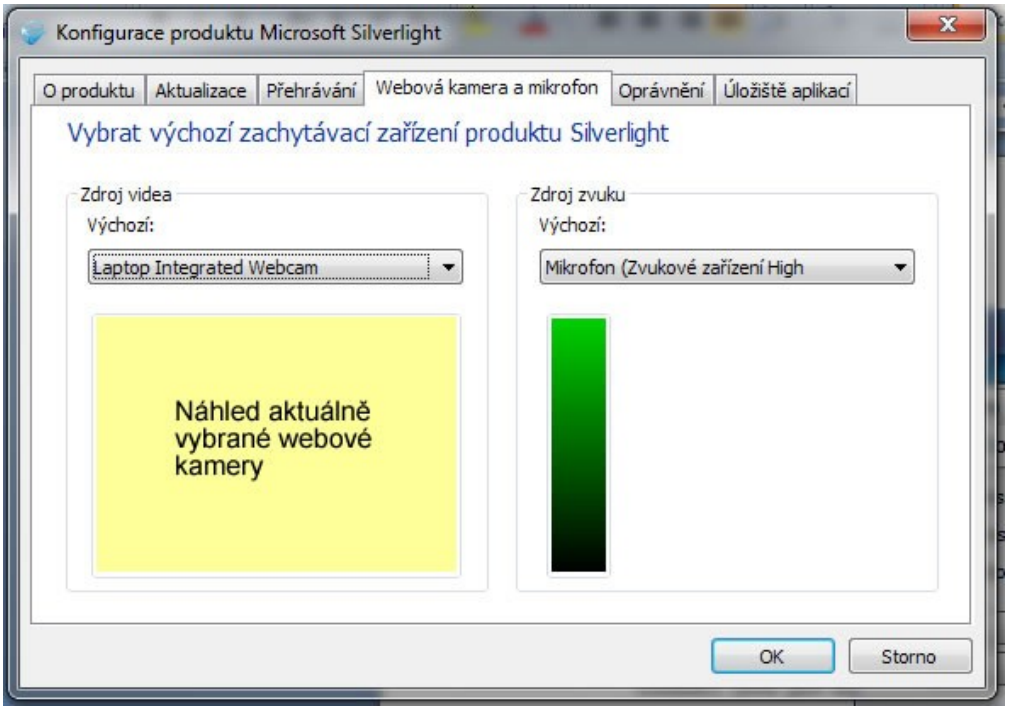
Pro každé zařízení je k dispozici také seznam formátů, které je možné využít. Tyto formáty jsou pro každé zařízení různé a dají se získat z kolekce SupportedFormats. Poté může aplikace uživateli nabídnout výběr jednoho z těchto formátů a vybraný formát se nastaví veřejné vlastnosti zařízení DesiredFormat. Tento přístup je vhodný hlavně tam, kde jsou zařízení použita ke komunikaci. Uživatel si může vybrat například formát s nižším rozlišením, pokud má nedostatečné připojení.

Příklad:

```
ReadOnlyCollection<VideoCaptureDevice> videoDevices =
    CaptureDeviceConfiguration.GetAvailableVideoCaptureDevices();

AvailableVideoDevices.ItemsSource = videoDevices;

//Uživatel vybírá zařízení

CaptureSource captureSource = new CaptureSource();

captureSource.VideoCaptureDevice =
    (VideoCaptureDevice)AvailableVideoDevices.SelectedItem;

ReadOnlyCollection<VideoFormat> availableFormats =
    captureSource.VideoCaptureDevice.SupportedFormats;

VideoDeviceFormats.ItemsSource = availableFormats;

//Uživatel vybírá formát

captureSource.VideoCaptureDevice.DesiredFormat =
    (VideoFormat)VideoDeviceFormats.SelectedItem;

//S vybraným zařízením se může pracovat
```

Konfigurace Silverlightu

Další možností je využívat defaultně nastavená zařízení. Výběr defaultního zařízení se provádí pomocí nástroje Konfigurace produktu Microsoft Silverlight. Tento konfigurační nástroj již byl popisován v kapitole o povolení přístupu k zařízením na straně klienta. Tento nástroj umožňuje uživateli vybrat z dostupných zařízení to zařízení, které chce defaultně využívat ve všech Silverlight aplikacích. Tato konfigurace je v nástroji dostupná v sekci Webová kamera a mikrofon. U každého zařízení je možné okamžitě sledovat náhled toho, co která webkamera snímá a u mikrofonu hlasitost snímaného zvuku.



Obrázek 15: Výběr defaultních zařízení

Když si uživatel vybere zařízení, které bude Silverlight považovat za defaultní, je možné k němu tímto způsobem přistupovat. To samozřejmě lze, i pokud defaultní zařízení uživatel nemění, protože v tom případě budou v konfiguračním nástroji vybrána zařízení považována za defaultní operačním systémem.

V Silverlight aplikacích se potom s těmito defaultními zařízeními pracuje podobně jako s výše zmíněnými zařízeními. Defaultní zařízení se opět získává pomocí statické třídy CaptureDeviceConfiguration, konkrétně se jedná o metody GetDefaultAudioCaptureDevice() a GetDefaultVideoCaptureDevice(). Tyto metody nevracejí kolekce dostupných zařízení, ale konkrétní zařízení, které je v konfiguračním nástroji Silverlightu nastaveno jako defaultní.

Příklad:

```
CaptureSource captureSource = new CaptureSource();

captureSource.VideoCaptureDevice =
    CaptureDeviceConfiguration.GetDefaultVideoCaptureDevice();

captureSource.AudioCaptureDevice =
    CaptureDeviceConfiguration.GetDefaultAudioCaptureDevice();

//S vybranými zařízeními se může pracovat
```

7 Ukázková aplikace

Jedním z cílů druhé části této práce je, pokud to technologie Silverlight dokáže, vytvořit aplikaci, která bude umožňovat video hovory. To znamená, že aplikace bude mezi klienty přenášet video i zvuk. Důvod pro vytvoření této aplikace je jednoduchý – ověřit v praxi všechny možnosti, zjištěné v předchozí části práce.

Prvním předpokladem pro to, aby bylo možné tuto aplikaci vytvořit je, že Silverlight umí pracovat se zařízeními na straně klienta. Dalším požadavkem je, aby s těmito zařízeními bylo možné pracovat tak, že aplikace dokáže zpracovávat audio a video data. Formát těchto dat také musí být možné v Silverlightu přehrávat, v nejlepším případě by tato data mělo být možné přehrávat v MediaElementu, který je pro tento účel přímo určen.

7.1 WCF Server

Aby bylo možné audio i video data posílat mezi uživateli používajícími tuto aplikaci, musí nějakým způsobem komunikovat. Silverlight stejně jako ostatní technologie z rodiny .NET umí snadno komunikovat s WCF službami. WCF služby proto byly nejjednodušší a nejlogičtější volbou pro komunikaci mezi klienty.

Takto vznikl server, ke kterému se klienti připojují a který jim umožní komunikaci mezi sebou. Tento server je pouze základním serverem, který klientovi umožňuje:

- Připojení k serveru
- Získání kolekce připojených klientů
- Odeslání audio a video dat na server – umožní ostatním uživatelům vidět obraz a zvuk
- Připojení k jinému klientovi
- Stahování audio a video dat od připojeného klienta

Server ale obsahuje řadu věcí, které by bylo vhodné vylepšit. Nejdůležitějšími nedostatky může být přihlašování uživatelů bez autentifikace a s tím související možnost stejných uživatelských jmen. To z hlediska bezpečnosti klientů a serveru není ideální. Dalším problémem může být, že není nijak vyřešeno odhlašování

uživatelů a tak mohou již nedostupní uživatelé působit jako připojení. Přesto tyto problémy nejsou vyřešeny. Důvodem pro takovéto řešení je, že popisovaná aplikace má předvést a otestovat možnosti Silverlightu a popisovaný server není určen pro veřejné používání, ale pouze pro to, aby aplikace mohla fungovat. Server není předmětem této práce, a proto vznikl pouze takto zjednodušeně.

Nejdůležitějšími částmi serveru jsou:

- Třída User
- Služba Service.svc

7.1.1 Třída User

Tato třída reprezentuje uživatele, který je přihlášen k serveru. Instance této třídy mají vlastní identifikátor a přezdívku, aby si uživatel mohl vybrat, s kým bude komunikovat. Dále instance této třídy uchovávají data, která jsou reprezentovaným uživatelem odesílána.

Tato třída, stejně jako zdrojový kód celého serveru, je k dispozici jako příloha. Samozřejmě včetně komentářů, které blíže popisují chování celé třídy a serveru.

7.1.2 Služba Service.svc

Toto je jediná služba, kterou server obsahuje a se kterou klienti mohou komunikovat. Třída obsahuje statickou kolekci uživatelů připojených k serveru. A samozřejmě metody, které klienti využívají ke komunikaci se serverem. Klientské aplikace umožňují přihlášení, spojení klientů, odesílání a stahování dat, odesílání a stahování použitého formátu. Tato služba poskytuje všechny metody, které jsou pro daný účel potřebné.

Opět je v příloze k dispozici okomentovaný zdrojový kód této služby a celého serveru.

7.1.3 Další třídy

Projekt WCF serveru obsahuje také další třídy, které jsou důležité, ale není potřeba je zvláště popisovat. Jejich účel i fungování je zřejmé z připojených komentářů a dokumentace.

7.2 Klientská aplikace

Klientská aplikace, která je kompletně vytvořena za pomoci technologie Silverlight, je hlavním tématem této kapitoly a bude popsána podrobněji. Aplikace nabízí výhody, které nabízí Silverlight, ovšem stejně tak je omezena některými nedostatky Silverlightu.

Tato aplikace samozřejmě musí obsahovat referenci na službu poskytovanou WCF serverem tak, aby mohla se serverem komunikovat.

Způsob, jak pracovat se zařízeními na straně klienta byl popisován v předchozích kapitolách a v této aplikaci jsou aplikovány právě tyto popisované postupy. V aplikaci se využívá tříd, které umožňují pracovat se zařízeními na straně klienta a zpracovávat data z těchto zařízení:

- VideoSink
- AudioSink

V těchto třídách se přistupuje přímo ke vzorkům z těchto zařízení, to znamená k audio vzorkům a snímkům z webkamery. Tyto vzorky jsou odesílány na server a jsou zde uchovávány jako aktuální snímek uživatele. Pokud jiný uživatel chce s tímto uživatelem komunikovat a slyšet zvuk a vidět video, které odesílá, musí si aktuální audio a video vzorky stahovat ze serveru.

Fungování těchto tříd je podobné fungování tříd popisovaných v předchozích kapitolách. Rozdíl je hlavně v tom, že data vzorků se nezapisují do paměti, ale odesílají se na server.

Jak už bylo v příslušných kapitolách zmíněno, tyto vzorky jsou syrové, Silverlight přistupuje přímo k syrovým audio i video datům. Aby bylo možné tato data v Silverlightu přehrávat, musí aplikace také obsahovat příslušnou implementaci třídy MediaStreamSource. Tato třída byla popisována v první části práce a je velice důležitou součástí aplikace. Konkrétní implementace využitá v aplikaci zpracovává audio data v syrovém formátu PCM a video data v syrovém formátu RGBA. Takto zpracovaná data je možné přehrávat v MediaElementu a tím zobrazit obraz i přehrát audio přenášené od připojeného uživatele.

V každém požadavku na snímek je snímek stažen ze serveru, zpracován a poskytnut MediaElementu k přehrání. Velkou výhodou tohoto přístupu je, že je synchronizován zvuk i obraz, protože každý vzorek (audio i video vzorky) je označen časovou známkou a tak je přehráván ve správný čas.

7.2.1 Fungování aplikace

Přihlášení k serveru

Uživatel aplikace se nejprve musí přihlásit k serveru, k tomu stačí zadat přezdívku. Také je možné se připojit jako skrytý uživatel, což je vhodné, pokud uživatel chce pouze přijímat video a audio a nechce vysílat. Tato možnost by mohla být využita například při video konferenci, kdy jeden uživatel vysílá a několik uživatelů přijímá. Další možností je připojit se pouze pro komunikaci mezi dvěma klienty, takže pokud se k uživateli připojí jiný klient, stává se uživatel nedostupným a nikdo jiný už se k němu připojit nemůže.

Po přihlášení k serveru má uživatel možnost získat kolekci dostupných uživatelů, má možnost začít odesílat data atd.

Připojení k uživateli

Po připojení je možné získat kolekci uživatelů, kteří jsou připojeni (není řešeno odpojování a tak mohou být již nedostupní). Z této kolekce si uživatel může vybrat, s kým chce komunikovat. Vybere si konkrétního uživatele a může se k němu připojit. Poté, co je uživatel připojený k jinému uživateli, je možné si od tohoto uživatele stahovat audio a video data, která jsou poté zobrazována.

Odesílání a přijímání dat

Data je možné nejen přijímat, ale i odesílat. Ovšem je možné si vybrat pouze jednu možnost a uživatel nemusí nutně odesílat i vysílat. Navíc, pokud uživatel nemá k dispozici webovou kameru nebo mikrofon, nebude video nebo audio data odesílat.

7.2.2 Výhody aplikace

Při vytváření aplikací tohoto typu v Silverlightu se nabízí několik zajímavých výhod. První z nich je to, že Silverlight je webová technologie, aplikace, kterou si uživatel může spustit všude, kde je Silverlight plugin nainstalovaný a je k dispozici internetové připojení pro přístup k této aplikaci. To znamená, že uživatel nemusí mít aplikaci nainstalovanou a může tuto aplikaci spouštět i tam, kde nemá oprávnění pro instalaci desktopových aplikací, například ve škole nebo v internetové kavárně.

Další výhodou je opačný případ, protože aplikace může být nainstalována na počítači uživatele a může být spouštěna přímo z plochy. To je vhodné pokud uživatel aplikaci využívá často a chce mít k aplikaci snadný přístup. To je možné díky tomu, že Silverlight umožňuje běh mimo prohlížeč – Out of Browser mód.

Další výhodou je již zmiňovaná synchronizace videa a audia. To je velice důležité při vytváření podobné aplikace a díky tomu je snadné vytvořit takovouto aplikaci v Silverlightu. Celkově je v Silverlightu (od verze 4) poměrně snadné pracovat se zařízeními na straně klienta a přistupovat k datům z těchto zařízení.

Také je velice užitečná možnost využít třídy odvozené z abstraktní třídy MediaStreamSource. Tím se vývojáři otevírá možnost přehrávat přenášená data tak, jako by se jednalo o jakékoliv jiné video nebo audio i ve formátu, jež v Silverlightu není standardně podporován.

7.2.3 Omezení a problémy aplikace

Hlavním problémem této aplikace při tvorbě v Silverlightu je to, že zpracovávaná data jsou syrová. To znamená, že zpracovávané vzorky jsou datově velmi náročné, to platí hlavně o video snímcích. Formát video dat je RGBA, to znamená, že na každý pixel snímku připadají 4 bajty udávající hodnotu těchto čtyř složek – červená, zelená, modrá a průhlednost. To znamená, že například při rozlišení snímků 640x480 má každý snímek velikost 1228800 bajtů (1200 kilobajtů). A když kamera poskytuje třicet snímků za vteřinu, znamená to, že by bylo potřeba zpracovávat přibližně 35 megabajtů za vteřinu.

Takové množství dat není možné v aplikaci zpracovávat. Proto je v aplikaci využito několik kroků, které tuto náročnost snižují:

- Na straně odesílajícího klienta je programově vybírán formát s nejnižším rozlišením, například místo 640x480 je vybrán formát 160x120 atd.
- Na server je odesílán jen každý druhý snímek – počet snímků za vteřinu je poloviční, než by bylo možné
- Ze serveru je stahován každý druhý snímek, počet snímků je při stahování snížen oproti serveru na polovinu – například když kamera poskytuje zmiňovaných třicet snímků za vteřinu, ze serveru jsou stahovány snímky jen 7,5x za vteřinu

Tato opatření jsou nutná pro fungování aplikace a je možné, že i přes tato opatření aplikace nebude fungovat ve všech podmínkách. Konkrétně problémy mohou nastat na slabších strojích a také tam, kde je využita kvalitní webová kamera, která například může poskytovat snímky ve vysokém rozlišení i v případě, kdy je vybráno nejnižší možné rozlišení. Jak je vidět, problémy se týkají hlavně přenosu videa, audio vzorky nemají takovou velikost a je možné je přenášet relativně bez problémů.

Pro řešení tohoto problému by bylo ideální využít nějakého kodeku, který by data uměl zakódovat a poté by bylo možné takovýto formát videa dekódovat a přehrát. V současné době není ale podobný kodek k dispozici a proto je nutné pracovat se syrovými daty.

To je největším problémem při tvorbě takovéto aplikace v Silverlightu. V C# není k dispozici žádný kodek, který by dokázal video data zakódovat. Takovýto kodek možná bude v budoucnosti vytvořen, ať už vývojáři využívajícími Silverlight nebo společností Microsoft v některé z dalších verzí. Zatím k dispozici není a tak je nutné s těmito omezeními počítat a k popisované aplikaci takto přistupovat. Z toho důvodu také není možné aplikaci využívat na internetu, ale je možné ji spouštět pouze lokálně.

7.2.4 Zdrojový kód

Okomentovaný zdrojový kód aplikace a také WCF serveru (tedy celého projektu), je k dispozici v příloze. Aplikace je funkční a je možné ji spustit a vyzkoušet, ovšem je nutné počítat s popisovanými omezeními.

8 Závěr

Cílem první části práce bylo popsat a ověřit možnosti Silverlightu při přehrávání streamovaného obsahu. Pro ověřování popisovaných možností byl navržen a proveden experiment. Pomocí popisovaného experimentu byl měřen výkon Silverlight aplikace a také zatěžování systému. Tento experiment dokázal, že Silverlight opravdu slibované možnosti dokáže nabídnout. Více k výsledkům je popsáno v příslušných kapitolách. Samozřejmě některé možnosti Silverlightu jsou závislé na použitém hardwaru a ne ve všech případech fungují ideálně. Konkrétně se jedná o přehrávání HD videa v rozlišení 1920x1080, které s integrovanou grafickou kartou nebylo plynulé.

Na základě výsledků tohoto experimentu lze v Silverlight aplikacích přehrávajících streamovaný obsah doporučit využívání GPU akcelerace. Používání GPU akcelerace snižuje zatížení systému a také zvyšuje výkon Silverlight aplikace. V případě, kdy poskytovatel obsahu má možnost poskytovat uživatelům video v HD kvalitě, měl by zvážit použití technologie Smooth Streaming. Z výsledků experimentu totiž vyplývá, že v současné době je přehrávání HD videa náročné na hardware koncového uživatele a na jeho připojení k síti. V takových případech je vhodné využít v práci popisovanou technologii Smooth Streaming.

Celkově Silverlight nabízí v oblasti multimédií (konkrétně videa a zvuku) široké možnosti, které uživatelům zpříjemňují využívání Silverlight aplikací a zjednodušují také použití Silverlightu pro poskytovatele obsahu. V neposlední řadě také Silverlight umožňuje programátorům, kteří takovéto aplikace vyvíjejí, jednoduché využívání těchto možností. Práce s těmito možnostmi je popsána v příslušných kapitolách.

Využití Silverlightu je v tomto oboru velmi široké. To je způsobeno také díky tomu, že Silverlight podporuje širokou škálu formátů videa a zvuku. Takže poskytovatelé obsahu mohou přejít na straně klienta na technologii Silverlight bez toho, aby museli jakkoliv měnit již vytvořený obsah. Navíc díky MediaStreamSource API je možné tuto širokou škálu podporovaných formátů rozšířit.

Abstraktní třída MediaStreamSource byla podrobně popsána a byl vytvořen i okomentovaný ukázkový příklad. Tento popis s příkladem může vývojářům velmi pomoci, protože na internetu se k tomuto tématu objevuje jen velice málo informací.

Rovněž bylo popsáno fungování Smooth Streamingu, který je na této abstraktní třídě založen a byly popsány shodné rysy mezi fungováním Smooth Streamingu a obecně všemi implementacemi třídy MediaStreamSource.

Ve druhé části práce byly nejprve popsány a předvedeny možnosti Silverlightu při práci s webkamerami a mikrofony. Protože se jedná o novinku v Silverlightu, tak byly popsány i základní postupy, například jak k těmto zařízením přistupovat. Ale byly samozřejmě popsány také pokročilejší postupy – jak zpracovávat data poskytovaná těmito zařízeními. K tomuto pokročilejšímu tématu také dosud není k dispozici mnoho zdrojů a tak ukázkové miniaplikace s okomentovanými zdrojovými kódy mohou posloužit ostatním programátorům.

Další důležitou součástí práce je ukázková aplikace, která využívá všech v práci zmíněných poznatků. Tato aplikace umožňuje video hovory – přenáší mezi uživateli aplikace video i zvuk. V této aplikaci bylo využito výhod, které v této oblasti Silverlight nabízí. Silverlight v této oblasti nenabízí pouze výhody, ale při vytváření takovéhoto typu aplikace vývojář narazí na jistá omezení a problémy. Tyto problémy se týkají především náročnosti takovéto aplikace, protože se pracuje se syrovými daty a Silverlight neposkytuje žádný kodek, který by tato data komprimoval do rozumného formátu. Proto byla využita jistá opatření, která snižují náročnost aplikace. Podrobněji k těmto nevýhodám a samozřejmě i výhodám viz příslušná kapitola, která je tomuto tématu věnována.

Hlavním závěrem, který z vytvoření této ukázkové aplikace vyplynul, je, že za použití Silverlightu je možné vytvořit tento typ aplikace. Ovšem je nutné počítat s popisovanými omezeními. Tato omezení, hlavně problémy plynoucí z práce s nekomprimovanými daty, jsou důvodem, proč v současnosti není Silverlight vhodný pro tento typ úloh. Chybějící kodek pro kódování syrových dat je Silverlightu vyčítán a tak se dá předpokládat, že tento nedostatek bude v některé z budoucích verzí Silverlightu odstraněn. Silverlight by se po odstranění tohoto nedostatku dal doporučit pro vytvoření aplikace umožňující video hovory.

Reference

[1] Flash Video: Progressive Download. *MediaCollege.com* [online]. c2010 [cit. 2010-02-06]. Dostupný z WWW: <http://www.mediacollege.com/adobe/flash/streaming/progressive-download.html>.

[2] Progressive download. *WhatIs.com* [online]. 2009-03-31 [cit. 2010-02-06]. Dostupný z WWW: <http://whatis.techtarget.com/definition/progressive-download.html>.

[3] Streaming or Progressive Download? *CoffeeCup* [online]. 2008-03-21 [cit. 2010-02-06]. Dostupný z WWW: <http://www.coffeecup.com/help/articles/streaming-or-progressive-download/>.

[4] Progressive download. *Wikipedia, the free encyclopedia* [online]. 2009-10-21 [cit. 2010-02-06]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Progressive_download>.

[5] Streaming media. *WhatIs.com* [online]. 2009-04-24 [cit. 2010-02-06]. Dostupný z WWW: <http://whatis.techtarget.com/definition/0,,sid9_gci753540,00.html>.

[6] Streaming video. *SearchUnifiedCommunications.com* [online]. 2009-08-06 [cit. 2010-02-06]. Dostupný z WWW: <http://searchunifiedcommunications.techtarget.com/sDefinition/0,,sid186_gci213055,00.html>.

[7] DUMLAO, Freedom. Silverlight Adaptive Streaming: How it works. *ASP.NET Weblogs* [online]. 2008-11-03 [cit. 2010-02-09]. Dostupný z WWW: <http://weblogs.asp.net/freedomdumlao/archive/2008/11/03/silverlight-adaptive-streaming-how-it-works.aspx>.

[8] ZAMBELLI, Alex. Smooth Streaming FAQ. *Alex Zambelli's Silverlight Media Blog* [online]. 2009-04-02 [cit. 2010-02-09]. Dostupný z WWW: <http://alexzambelli.com/blog/smooth-streaming-faq/>.

[9] ZAMBELLI, Alex. A Brief History of Multi-Bitrate Streaming. *Alex Zambelli's Silverlight Media Blog* [online]. 2009-12-17 [cit. 2010-02-09]. Dostupný z WWW: <http://alexzambelli.com/blog/2008/12/17/a-brief-history-of-multi-bitrate-streaming/>

[10] KNOWLTON, Chris. Adaptive Streaming Comparison. *IIS.net* [online]. 2010-01-28 [cit. 2010-02-09]. Dostupný z WWW: <http://learn.iis.net/page.aspx/792/adaptive-streaming-comparison>.

[11] Compatible Operating Systems and Browsers. *Microsoft.com* [online]. c2010 [cit. 2010-02-13]. Dostupný z WWW: <http://www.microsoft.com/silverlight/get-started/install/default.aspx>.

[12] Server-Side Playlists. *MSDN* [online]. c2010 [cit. 2010-02-15]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/cc645037(VS.95).aspx>.
[13] Audio and Video Overview. *MSDN* [online]. c2010 [cit. 2010-02-13]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/cc189078(VS.95).aspx>.
[14] Supported Media Formats, Protocols, and Log Fields. *MSDN* [online]. c2010 [cit. 2010-02-13]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/cc189080(VS.95).aspx>.
[15] Digital Rights Management (DRM). *MSDN* [online]. c2010 [cit. 2010-02-13]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/cc838192(VS.95).aspx>.
[16] Media Technical Details. *Microsoft.com* [online]. c2010 [cit. 2010-02-13]. Dostupný z WWW: <http://www.microsoft.com/silverlight/overview/media/media-details.aspx>.
[17] WAGGONER, Ben. Silverlight 3 Beta - What's New for Media. *Channel 10* [online]. 2009-03-18 [cit. 2010-02-13]. Dostupný z WWW: <http://on10.net/blogs/benwagg/Silverlight-3-Beta-Whatrsquos-New-for-Media/>.
[18] WAGGONER, Ben. H.264 and AAC support coming in Silverlight. *Channel 10* [online]. 2008-09-09 [cit. 2010-02-13]. Dostupný z WWW: <http://on10.net/blogs/benwagg/H264-and-AAC-support-coming-in-Silverlight/>.
[19] WAGGONER, Ben. Media Technologies in Silverlight: Windows Media and VC-1. *Microsoft Expression* [online]. 2008-04 [cit. 2010-02-13]. Dostupný z WWW: <http://www.microsoft.com/expression/news-press/newsletter/2008-04/Article02.aspx>.
[20] MediaElement.Markers Property. *MSDN* [online]. c2010 [cit. 2010-02-13]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/system.windows.controls.mediaelement.markers(VS.95).aspx>.
[21] Qumu Delivers New Multicast Plug-In for Microsoft Silverlight. *Qumu* [online]. 2009-05-11 [cit. 2010-02-13]. Dostupný z WWW: <http://www.qumu.com/newsroom/news-releases/2009/silverlightplugin.htm>.
[22] Multicast Comes to Silverlight!. *The Silverlight Blog* [online]. 2009-05-11 [cit. 2010-02-13]. Dostupný z WWW: <http://team.silverlight.net/announcement/multicast-comes-to-silverlight/>.
[23] Project Starlight: Multicast Plug-in for Silverlight. *The Official Microsoft IIS Site* [online]. 2009-05-11 [cit. 2009-02-13]. Dostupný z WWW: <http://blogs.iis.net/benwagg/archive/2009/05/11/project-starlight-multicast-plug-in-for-silverlight.aspx>.

[24] MediaElement.Source Property. *MSDN* [online]. c2010 [cit. 2010-02-23]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/system.windows.controls.mediaelement.source(VS.95).aspx>.
[25] MediaElement.SetSource Method (Stream). *MSDN* [online]. c2010 [cit. 2010-02-23]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/cc190669(VS.95).aspx>.
[26] MediaElement.SetSource Method (MediaStreamSource). *MSDN* [online]. c2010 [cit. 2010-02-23]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/cc626563(VS.95).aspx>.
[27] KHOUZAM, Gilles. Playing back Wave files in Silverlight. *Gilles' Thoughts* [online]. 2009-03-23 [cit. 2010-02-23]. Dostupný z WWW: <http://blogs.msdn.com/gillesk/archive/2009/03/23/playing-back-wave-files-in-silverlight.aspx>.
[28] KHOUZAM, Gilles. WaveMediaStreamSource. *MSDN* [online]. 2009-03-23 [cit. 2010-02-23]. Dostupný z WWW: <http://code.msdn.microsoft.com/wavmss>.
[29] Downloading Content on Demand. *MSDN* [online]. c2010 [cit. 2010-02-23]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/cc189021(VS.95).aspx>.
[30] BROWN, Pete. Creating Sound using MediaStreamSource in Silverlight 3 Beta. *POKE 53280,0: Pete Brown's Blog* [online]. 2009-03-2009 [cit. 2010-02-25]. Dostupný z WWW: <http://community.irritatedvowel.com/blogs/pete_browns_blog/archive/2009/03/23/Creating-Sound-using-MediaStreamSource-in-Silverlight-3-Beta.aspx>.
[31] MediaElement.SetSource Method (MediaStreamSource). *MSDN* [online]. c2010 [cit. 2010-02-23]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/cc626563(VS.95).aspx>.
[32] WAGGONER, Ben. Silverlight 3 Beta - What's New for Media. *Channel 10* [online]. 2009-03-18 [cit. 2010-02-13]. Dostupný z WWW: <http://on10.net/blogs/benwagg/Silverlight-3-Beta-Whatrsquos-New-for-Media/>.
[33] WAGGONER, Ben. First MediaStreamSource example is up. *Channel 10* [online]. [online]. 2008-10-01 [cit. 2010-02-25]. Dostupný z WWW: <http://on10.net/blogs/benwagg/First-MediaStreamSource-example-is-up/>.
[34] BROWN, Pete. Silverlight Synthesizer Source Code – MediaStreamSource Raw Sound. *POKE 53280,0: Pete Brown's Blog* [online]. 2009-07-19 [cit. 2010-02-25]. Dostupný z WWW: <http://community.irritatedvowel.com/blogs/pete_browns_blog/archive/2009/07/19/Silverlight-Synthesizer-Source-Code-_1320_-MediaStreamSource-Raw-Sound.aspx>.

[35] BROWN, Pete. Silverlight 3 – Creating Video from Raw Bits using a MediaStreamSource. *POKE 53280,0: Pete Brown's Blog* [online]. 2009-03-18 [cit. 2010-02-25]. Dostupný z WWW: <http://community.irritatedvowel.com/blogs/pete_browns_blog/archive/2009/03/18/Silverlight-3-_1320_-Creating-Video-from-Raw-Bits-using-a-MediaStreamSource.aspx>.

[36] ZAMBELLI, Alex. MediaStreamSource Takes On a New Life. *Alex Zambelli's Silverlight Media Blog* [online]. 2009-03-26 [cit. 2010-02-25]. Dostupný z WWW: <http://alexzambelli.com/blog/2009/03/26/mediastreamsource-takes-on-a-new-life/>.

[37] PETZOLD, Charles. A Simple Electronic-Music Sequencer for Silverlight 3. *Petzold Book Blog* [online]. 2009-07-12 [cit. 2010-02-25]. Dostupný z WWW: <http://www.charlespetzold.com/blog/2009/07/Simple-Electronic-Music-Sequencer-for-Silverlight.html>.

[38] MediaStreamSource Class. *MSDN* [online]. c2010 [cit. 2010-02-25]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/system.windows.media.mediastreamsource(VS.95).aspx>.

[39] MediaStreamSample Class. *MSDN* [online]. c2010 [cit. 2010-02-25]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/system.windows.media.mediastreamsample(VS.95).aspx>.

[40] MediaStreamDescription Class. *MSDN* [online]. c2010 [cit. 2010-02-25]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/system.windows.media.mediastreamdescription(VS.95).aspx>.

[41] NAB09 - Microsoft Releases IIS Smooth Streaming for True HD 1080p Video Delivery. *The Silverlight Team Blog* [online]. 2009-04-23 [cit. 2010-02-25]. Dostupný z WWW: <http://team.silverlight.net/events/nab09-microsoft-releases-iis-smooth-streaming-for-true-hd-1080p-video-delivery/>.

[42] MediaStreamSource.GetSampleAsync Method. *MSDN* [online]. c2010 [cit. 2010-02-25]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/system.windows.media.mediastreamsource.getsampleasync(VS.95).aspx>.

[43] MediaStreamType Enumeration. *MSDN* [online]. c2010 [cit. 2010-02-25]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/system.windows.media.mediastreamtype(VS.95).aspx>.

[44] ZAMBELLI, Alex. Smooth Streaming FAQ. *Alex Zambelli's Silverlight Media Blog* [online]. 2009-04-02 [cit. 2010-03-07]. Dostupný z WWW: <http://alexzambelli.com/blog/smooth-streaming-faq/>.

[45] ZAMBELLI, Alex. IIS Smooth Streaming Technical Overview. *Microsoft Corporation* [online]. 2009-03-25 [cit. 2010-03-07]. Dostupný z WWW:

<http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=03d22583-3ed6-44da-8464-b1b4b5ca7520>.

[46] IIS Media Services. *The Official Microsoft IIS Site* [online]. c2010 [cit. 2010-03-07]. Dostupný z WWW: <http://www.iis.net/media>.

[47] ZAMBELLI, Alex. Smooth Streaming Architecture. *Alex Zambelli's Silverlight Media Blog* [online]. 2009-02-10 [cit. 2010-03-07]. Dostupný z WWW: <http://alexzambelli.com/blog/2009/02/10/smooth-streaming-architecture/>.

[48] Live Smooth Streaming. *The Official Microsoft IIS Site* [online]. c2010 [cit. 2010-03-07]. Dostupný z WWW: <http://www.iis.net/expand/LiveSmoothStreaming>.

[49] DOWNEY, Mike. Silverlight Media Framework. *CodePlex* [online]. 2010-03-01 [cit. 2010-03-07]. Dostupný z WWW: <http://smf.codeplex.com/>.

[50] IIS Smooth Streaming Player Development. *MSDN* [online]. c2010 [cit. 2010-03-07]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/ee958035.aspx>.

[51] EnableFramerateCounter (Silverlight Plug-in Object). *MSDN* [online]. c2010 [cit. 2010-03-07]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/cc838171(VS.95).aspx>.

[52] EnableGPUAcceleration (Silverlight Plug-in Object). *MSDN* [online]. c2010 [cit. 2010-03-07]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/dd833062(VS.96).aspx>.

[53] CaptureSource Class. *MSDN* [online]. c2010 [cit. 2010-03-22]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/system.windows.media.capturesource(VS.96).aspx>.

[54] VideoBrush Class. *MSDN* [online]. c2010 [cit. 2010-03-22]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/system.windows.media.videobrush(VS.95).aspx>.

[55] AudioSink Class. *MSDN* [online]. c2010 [cit. 2010-03-22]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/system.windows.media.audiosink(VS.96).aspx>.

[56] VideoSink Class. *MSDN* [online]. c2010 [cit. 2010-03-22]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/system.windows.media.videosink(VS.96).aspx>.

[57] CaptureDeviceConfiguration Class. *MSDN* [online]. c2010 [cit. 2010-03-22]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/system.windows.media.capturedeviceconfiguration(VS.96).aspx>.

[58] TAULTY, Mike. Silverlight 4 Rough Notes: Camera and Microphone Support Class. *Mike Taulty's Blog* [online]. 2009-11-18 [cit. 2010-03-22]. Dostupný z WWW:

<http://mtaulty.com/CommunityServer/blogs/mike_taultys_blog/archive/2009/11/18/silverlight-4-rough-notes-camera-and-microphone-support.aspx>.

[59] Browser Statistics. *W3schools.com* [online]. c2010 [cit. 2010-03-31]. Dostupný z WWW: <http://www.w3schools.com/browsers/browsers_stats.asp>.

[60] MARCHAL, Chris. Using a webcam in WPF. *MediaMash* [online]. 2006-08-25 [cit. 2010-03-19]. Dostupný z WWW: <http://blogs.msdn.com/cmarchal/archive/2006/08/25/724224.aspx>.

[61] MORRILL, Jeremiah. WPF MediaKit - For webcam, DVD and custom video support in WPF. *CodePlex* [online]. 2009-10-11 [cit. 2010-03-19]. Dostupný z WWW: <http://www.codeplex.com/WPFMediaKit>.

[62] Silverlight 4 Information. *Silverlight.NET* [online]. c2010 [cit. 2010-03-19]. Dostupný z WWW: <http://silverlight.net/getstarted/silverlight-4/>.

[63] About Windows Media High Definition Video. *Microsoft Windows Media* [online]. c2010 [cit. 2010-03-27]. Dostupný z WWW: <http://www.microsoft.com/windows/windowsmedia/default.mspx>.