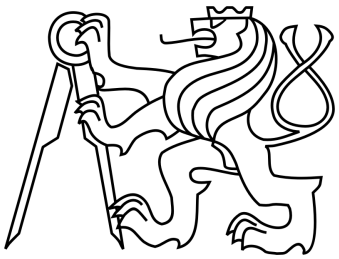
České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra radioelektroniky



BAKALÁŘSKÁ PRÁCE

Interaktivní multimediální aplikace

Interactive Multimedia Application

Autor: Tomáš Suchomel

Vedoucí práce: Ing. Stanislav Vítek, Ph.D.

Praha, 2015

Rád bych tímto poděkoval panu Ing. Stanislavovi Vítkovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích během vypracování bakalářské práce.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů v souladu s Metodickým pokynem č. 1/2009. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 22. května 2015 .....

Podpis

Název práce: Interaktivní multimediální aplikace

Autor: Tomáš Suchomel

Katedra: Katedra radioelektroniky

Vedoucí bakalářské práce: Ing. Stanislav Vítek, Ph.D.

Abstrakt Tato bakalářská práce se zabývá návrhem interaktivní multimediální aplikace, která umožňuje uživateli ovlivnit podobu živého video přenosu. Aplikace spojuje možnosti přehrávacího programu CasparCG a sociální sítě Twitter. V rámci práce byl vytvořen program interpretující příspěvky uživatele sociální sítě, označený specifickým atributem (tzv. hashtagem), jako příkazy protokolu ovládajícího streamující program CasparCG.

Klíčová slova: Interaktivita, multimediální aplikace, Twitter API, CasparCG, Stream

---

Title: Interactive multimedia application

Author: Tomáš Suchomel

Department: Department of Radioelectronics

Supervisor: Ing. Stanislav Vítek, Ph.D.

Abstract This bachelor thesis deals with the design of an interactive multimedia application, which allows the user to influence the form of a live video stream. The application connects the possibilities of the CasparCG playout application and the Twitter social network. A program interpreting a post from a social network user, labeled by a specific attribute (so called hashtag), as a command of a protocol that controls the CasperCG streaming program, was developed within this work.

Keywords: Interactivity, multimedia apllication, Twitter API, CasparCG, Stream

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra radioelektroniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Tomáš Suchomel**

Studijní program: Komunikace, multimédia a elektronika
Obor: Multimediální technika

Název tématu: **Interaktivní multimediální aplikace**

Pokyny pro vypracování:

1. Navrněte interaktivní multimediální aplikaci, která umožní změnu multimediálního obsahu (videostream) v reálném čase na základě požadavku uživatele aplikace.
2. Požadavek je možné zadat pomocí prostředků sociálních sítí, jako je Twitter, případně Facebook.
3. Aplikace implementujte pomocí streamovacího serveru CasparCG.

Seznam odborné literatury:

[1] New Directions in Intelligent Interactive Multimedia Systems and Services - 2 (Studies in Computational Intelligence), Damiani E., Jeong J (Editors), Studies in Computational Intelligence (Book 226), Springer, 2009, ISBN 978-3642029363
[2] CasparCG Documentation Wiki [online], http://casparcg.com/wiki/Main_Page

Vedoucí: Ing. Stanislav Vítek, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Mgr. Petr Páta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 2. 2015

OBSAH

KAPITOLA

PRVNÍ

ÚVOD

Tato bakalářská práce se zabývá možnostmi zvýšení interaktivity při živých přenosech multimediálních informací s využitím možností, které poskytují sociální sítě jako je Twitter nebo Facebook. Interaktivita je zde chápána jako prostředek pro zvýšení aktraktivity přenosu tím, že je divákovi umožněno pomocí určitého komunikačního kanálu ovlivňovat parametry a průběh přenosu a podílet se tak na jeho vývoji. Díky tomu roste jeho požitek a zájem o sledování přenosu.

S myšlenkou interaktivního filmu přišel poprvé Radúz Činčera v 60. letech minulého století. Jednalo se o projekt Kinoautomat s filmem Člověk a jeho dům, který měl svou premiéru během světové výstavy EXPO 1967 v Montrealu zvané „Man and his World“. Diváci sledovali příběh pana Nováka, ztvárněného Miroslavem Horníčkem, a v klíčových okamžicích děje měli možnost ovlivnit hlasováním rozhodnutí hlavního hrdiny. Dějové linie však z celkem pochopitelných důvodů po nějakém čase opět splývají do jedné. Technická realizace projektu byla samozřejmě nesmírně náročná, avšak tvůrci za něj sklidili nečekaný ohlas. Vzhledem k totalitnímu systému v tehdejším Československu bylo však promítání filmu po nějaké době zakázáno a postupem času upadl tento obří a nadčasový projekt v zapomnění.[1],[2]

Myšlenka aktivního zapojení diváka však zůstala zachována a nabyla na významu s příchodem internetové televize a příbuzných technologií. Diváci tak mají například možnost během přenosu zápasu získávat informace o jednotlivých hráčích. Prosadil se model Video on Demand (VoD), kdy má divák možnost přehrát libovolný multimediální obsah. Tento model je nově rozšiřován také pomocí technologie HbbTV - Hybrid Broadcast Broadband TV, která kombinuje vysílání

s širokopásmovým internetem.

Myšlenka, která je jádrem této bakalářské práce, je do značné míry ovlivněna již zmíněným Kinoautomatem, či podobným modelem, jako je např. Jukebox. Diváci, kteří disponují vhodným komunikačním kanálem, mohou ovlivnit průběh živého videostreamu - vkládat doplňkové informace, hlasovat v soutěži a podobně. Jako vhodný komunikační kanál se jeví sociální služba Twitter, která kromě posílání krátkých textových zpráv přinesla i tzv. hashtagy, umožňující seskupovat příspěvky podle jejich tématu. Kromě toho disponuje služba velmi dobře navrženým API, které umožňuje jednoduchou práci se zprávami.

Práci jsem strukturoval následujícím způsobem: druhá kapitola je věnována programu CasparCG. Tento program, napsaný programátory Švédské státní televize SVT, je volně šiřitelný playout software určený především pro televizní vysílání. Program by se v základu dal přirovnat ke klasické hardwarové video-střižně, kterou je však možné libovolně programovat. Navíc oproti klasické střižně má tento program řadu funkcí, které jsou neustále rozšiřovány jak autory samotného programu, tak nadšenými uživateli, které program používají pro rozličné účely. Mezi hlavní výhody programu patří podpora flashových šablon, které umožňují jednoduché titulkování videa či přehrávání animací, práce jak s video vstupy grafické karty, tak s daty uloženými na disku počítače a v neposlední řadě také nově přidaná možnost konfigurace výstupu jako stream na internet pomocí vestavěného modulu FFmpeg. Program samotný je pouze server bežící na určitém portu, na který lze přistupovat pomocí telnetu a řídit ho pomocí specifických příkazů. Díky tomu lze vytvořit vlastního telnetového klienta, kterému lze naprogramovat libovolné funkce, a tak například zautomatizovat průběh střihu.

Ve třetí kapitole jsou popsány již zmíněné funkce API Twitteru a přístup k nim. Díky nim je například možné v reálném čase filtrovat a stahovat příspěvky ostatních uživatelů zvané tweety. Jsou zde popsány jak možnosti datových typů, tak jejich struktura.

Čtvrtá kapitola je věnována telnetovému klientovi BPstream. Klient propojuje všechny části této aplikace, od získávání a stahování tweetů, přes samotnou práci s daty a jejich vyhodnocování, po řízení programu CasparCG. Tento klient byl napsán v programovacím jazyku Python, který se pro tento účel jevil jako nejvhodnější varianta.

Pátá kapitola popisuje závěrečné implementování aplikace do celého systému, včetně samotné konfigurace streamu a jeho výsledného vzhledu.

KAPITOLA

DRUHÁ

CASPARCG

Program CasparCG, profesionální grafický a play-out software, byl vyvinut roku 2006 programátory švédské státní televize SVT[1], která ho od té doby využívá k nepřetržitému broadcastu. Celý systém stojí na programu CasparCG Server, který nemá své vlastní grafické uživatelské rozhraní, ale lze na něj přistupovat pomocí telnetu.

Vzhledem k tomu, že byl napsán jako Open Source s licencí GNU GPLv3[2], je volně dostupný i programátorům, kteří ho chtějí dále vyvíjet a vytvářet si nástavby přímo pro své účely. Díky této komunitě vývojářů obsahuje program veliké množství modulů, které mají různorodé funkce, a proto má program velmi široké možnosti použití. Program jako takový by se dal přirovnat ke klasické hardwarové video-střižně, kde ale na vstup lze nakonfigurovat kromě zdrojů připojených do grafické karty i různorodá data uložená v počítači, dokonce i odchytávaný stream z internetu. Mezi data uložená v počítači mohou patřit jak videa, obrázky, tak různé šablony. Vše včetně podpory alfa-kanálu. Program umí pracovat jak s HTML kódem, tak flashovými šablony, které mohou mít oproti HTML šablonám i animovaný obsah. Veškeré vstupy se rozdělí do jednotlivých vrstev, které lze různě transformovat, přidávat barevné i jiné filtry, nebo je klíčovat. Výsledkem je obraz složený překrýváním těchto vrstev.

Existuje mnoho způsobů, jak a kam výsledný obraz exportovat. Stejně jako u hardwarové video-střižny může být obraz samozřejmě posílán na výstup grafické karty, stejně tak ale lze obraz například streamovat na internet. Od verze

[1] www.svt.se

[2] http://casparcg.com/wiki/CasparCG_Server#License

CasparCG Server 2.0.7 podporuje program i FFmpeg, proto lze přímo v rámci programu video konvertovat do formátu H264 a následně ho posílat na nějaký streamovací server. To je možnost, která je využita v rámci této bakalářské práce.

Následující kapitola je sepsána podle dostupných informací na stránkách autora a tam umístěného fóra.[3]

2.1 Instalace

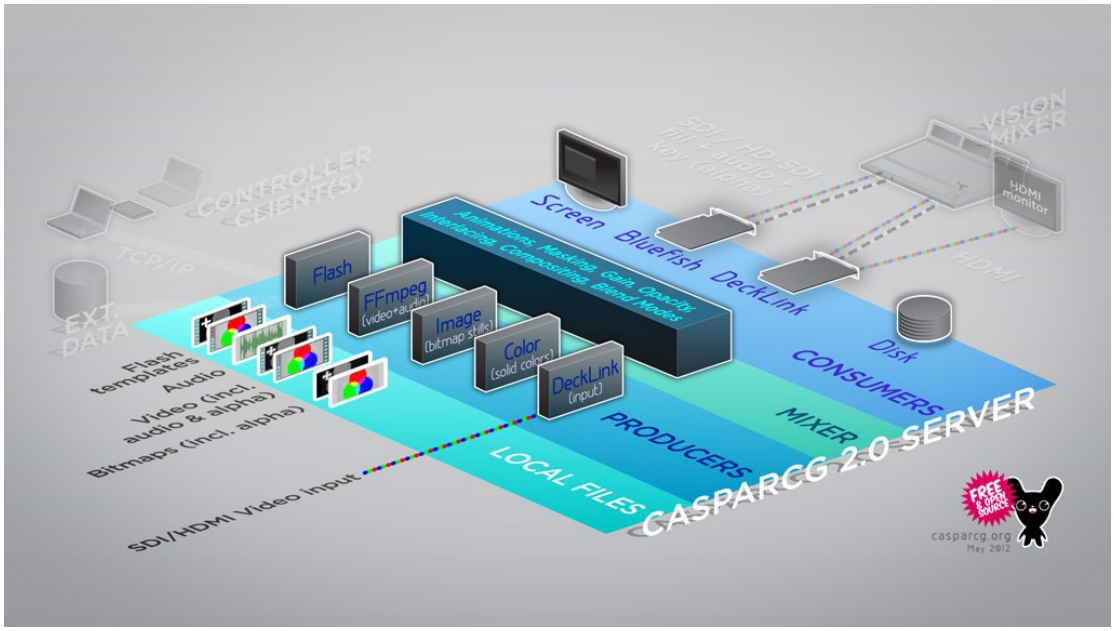
Na webových stránkách produktu[3] je ke stažení samotný program CasparCG Server a dále několik doplňků, jako například generátor flashových šablon Generate Template pro Adobe Flash Professional a Adobe After Effect. Jak již bylo zmíněno v úvodu kapitoly, program je v podstatě server běžící na daném portu a lze ho řídit pomocí telnetu. Z tohoto důvodu je ke stažení i CasparCG Client, telnetový klient s grafickým rozhraním, demonstrujícím hlavní funkce programu.

Na samotné instalaci není nic složitého – jedná se o přenosný program, tzv. portable, tudíž pokud jsou dodrženy systémové požadavky, mělo by vše proběhnout bez sebemenších problémů. Pokud však přeci jen nějaký problém nastane, téměř vždy lze na něj nalézt řešení na již zmíněném fóru.

2.2 CasparCG Server

Jak je znázorněno na obrázku 2.1, server pracuje na třech hlavních vrstvách - modulech, a to Producers, Mixer a Consumer. Jako čtvrtý modul, podle obrázku první v pořadí, by se dala považovat vrstva řídící lokální soubory – vrstva Local Files. Server umí pracovat s audiem, videem, obrázkovými soubory uloženými na disku počítače nebo například flashovými šablonami, jejichž schopnost přehrávání a následném titulkování v reálném čase je jednou z největších předností serveru. První hlavní vrstvou je vrstva PRODUCERS, která má za úkol pracovat se vstupními daty, mezi které lze zařadit jak data a média uložená na počítači, tak i například streamované video nebo vstup z video-karty. Modul MIXER zajišťuje kombinování jednotlivých vrstev vytvořených modulem Producers za pomoci různých filtrů a funkcí pro transformace obrazu. Mohou zde tak probíhat animace přechodů, klíčování vstupů a podobně. Modul CONSUMERS umožňuje vytvořit mnoho typů výstupů, jako například výstup na monitor ve formátu PAL, výstup

[3] www.casparcg.com/download



Obrázek 2.1: Schéma modulů serveru

na video-střižnu DeckLink, BlueFish nebo ATEM a v neposlední řadě také ukládat výstup na disk počítače nebo ho streamovat na internet.

Konfigurace vstupů, výstupů a podobných neměnných parametrů se doporučuje dělat ještě před spuštěním serveru, a to v XML souboru casparcg.config umístěném v kořenové složce programu. Při každé změně původní konfigurace je třeba celý server restartovat. Většina nastavení však lze měnit i za běhu serveru pomocí telnetu.

2.2.1 Základní příkazy

Veškerá komunikace probíhá prostřednictvím protokolu AMCP. Ten disponuje velikým množstvím příkazů, ze kterých je zde uvedena pouze nejdůležitější část pro tuto práci. Každý příkaz se skládá z několika částí: názvu příkazu, čísla kanálu a vrstvy, na kterou je příkaz směřován a dále doplňující parametry, které jsou závislé na typu příkazu.

>>PRIKAZ kanal-vrstva další parametry

Prvním takovým příkazem je **LoadBG**, díky kterému lze načíst jakýkoliv zdroj na pozadí programu a připravit ho tak pro rychlé spuštění. Příkaz má také široké možnosti nastavení. Pomocí předem definovaného tvaru zápisu lze nastavit nejen informace o zdroji - jeho adresu, kanál a vrstvu, do které bude umístěn. Příkaz na-

bízí také doplňující možnosti nastavení, jako například zda bude video přehráváno ve smyčce, jaký bude typ přechodu a délku jeho trvání, typ časového průběhu animace tween přechodu 2.2.2 včetně toho, na jakou stranu bude přechod orientován. Poslední parametry nastavují takzvaný *seek*, spuštění videa až od určitého framu, dále délku videa ve framech, možnost *auto* - přehrání videa automaticky poté, co se daná vrstva uvolní, a na závěr lze nastavit možnost použití takzvaného Libav filtru. Seznam všech podporovaných typů filtru je na tomto odkaze[4].

Další příkaz se jmenuje **Load**. Jedná se o jednodušší příkaz, který pouze nahraje video (nebo jakýkoli zdroj) do dané vrstvy, zobrazí ho a zastavené video dále čeká na spuštění pomocí funkce Play. Nelze pak však nastavit doplňující parametry jako u LoadBG.

Zde je porovnání obou příkazů:

>>LOADBG 1-10 nazev_souboru PUSH 20 easeinesine LOOP SEEK 200 LENGTH 400 AUTO FILTER hflip

>>LOAD 1-10 nazev_souboru

Příkaz **Play** aktivuje danou vrstvu, a pokud má definovanou i adresu zdroje, přehraje jej. Má však tu nevýhodu, že při zadání se musí video jak načíst, tak spustit. To může mít za následek jisté časové zpoždění, které pak vytváří problém u sekvencí s přesně nastaveným časem přehrávání. Proto je vhodné zdroj načíst se značným předstihem pomocí příkazu LoadBG a poté vrstvu pouze aktivovat příkazem Play. Vzhledem k tomu, že funkce LoadBG umožňuje nahrát zdroj na vrstvu bez přerušení aktuálního obrazu vrstvy, dochází při kombinaci těchto příkazů k výrazně nižšímu časovému zpoždění.

Příkaz Play se tedy používá pro dva případy:

>>PLAY 1-10

>>PLAY 1-10 nazev_souboru

Příkaz **Pause** umožňuje pozastavit obraz dané vrstvy na aktuálním framu. Zastavený obraz se dá obnovit buď příkazem Play anebo přímo pro to určeným příkazem **Resume**. Pro zastavení a ukončení aktuálně přehrávaného videa je určen příkaz **Stop**. Při použití tohoto příkazu tak zůstává zachován veškerý obsah načtený pomocí příkazu Load či LoadBG. Tento obsah spolu s aktuálně přehrávaným lze smazat příkazem **Clear**, který je definován buď pro danou vrstvu, nebo pro celý kanál. Veškerá nastavení vrstvy pomocí modulu MIXER však zůstává při použití těchto příkazů zachováno. Pro vymazání těchto nastavení slouží příkaz

[4]http://libav.org/libavfilter.html

MIXER Clear, který je zmíněn v podkapitole číslo 2.2.2.

Tyto příkazy mohou vypadat následovně.

>>PAUSE 1-10

>>RESUME 1-10

>>STOP 1-10

>>Clear 1-10

>>Clear 1

2.2.2 Moduly programu

Local Files

Modul spravující data uložená v počítači, angl. Local Files, zpracovává data uložená v kořenové složce. Ta je defaultně nastavená do kořenové složky celého programu. V konfiguračním souboru popsaném v kapitole 2.2.3 na straně 14 je však možné tuto adresu složky změnit.

Producers

Flash Producer Prvním modulem ve vrstvě Producers, je Flash modul. Podporované formáty jsou .ft – Flash template, jde v podstatě o SWF formát, který byl vygenerovaný pomocí generátoru šablon Template Generator, který lze nainstalovat přímo do programu, kde se flashová šablona vytváří – například Adobe Flash Pro nebo Adobe After Effect. Dalším formátem může být .ct, jehož obsahem je společně flashová šablona s medii a XML daty. Posledním typem je samozřejmě klasický .swf formát. Všechny soubory, které má Flash modul přehrát, musí být uloženy v kořenových složkách.

FFmpeg Producer Další modul, FFmpeg Producer, řídí vstup všech audio-video médií, které lze přehrát pomocí FFmpeg a QuickTime video kodeků, jako například PNG, JPEG, Motion JPEG, H.264, FLV, WMV a dalších včetně nekomprimovaného videa. Dále je možnost na vstup nakonfigurovat živý video-stream například z youtube nebo VLC Playeru. Samozřejmostí je podpora alfa-kanálu. Tyto soubory lze pomocí jednoduchých příkazů spouštět a dále s nimi pracovat.

Všechna média musí být uložena v kořenové složce serveru – typicky ve složce media. Nahrání a spuštění se provede příkazem:

>> PLAY 1-10 MOVIE LENGTH 200 LOOP

Kde MOVIE znamená název média, lze ho zadávat bez přípony. Dále je možné přidat doplňující nastavení, například LENGTH pro délku vyjádřenou ve framech a LOOP, které nastavuje přehrávání ve smyčce.

Image Producer Tento modul je podobný předešlému, pouze zaměřen na zpracování obrázků obvyklých formátů. Obrázky nejsou škálovány, proto je třeba, aby se jejich rozměr shodoval s rozlišením kanálu.

Color Producer Color Producer generuje barvu pozadí výstupu kanálu. Nastavuje se příkazem >>PLAY 1 #FFFFFF, kde číslo 1 udává číslo kanálu a hexadecimální číslo barvu. Tento příklad je pro bílé pozadí.

DeckLink Producer Modul pro odchytávání videa ze vstupu grafické karty DeckLink. Na tento vstup může být připojen například výstup z klasické hardwarové video-střižny, do kterého CasparCG následně vkládá animace, titulky, flashové šablony a podobně.

Mixer

Tato vrstva je v podstatě samotný modul, který řídí všechny operace typu animace, skládání vrstev, klíčování a podobně. Všechna média jsou skládána/vrstvena podle předem definovaného pořadí. Číslo dané vrstvy se definuje již v příkazu pro její načtení do serveru. Pokud je u dané vrstvy (videa, obrázku nebo flashové šablony) přiložen alfa-kanál, server ho samozřejmě zohledňuje a dojde k prolínání vrstvy s vrstvou nižšího čísla. Mixer modul zároveň dokáže v reálném čase (při použití výkonnějšího grafického procesoru) tyto vrstvy přeskupovat a dále je transformovat.

Anchor, Fill Jednou z možností transformování obrazu, a asi tou nejčastěji využívanou, je nastavení polohy a velikosti obrazu. Od verze 2.0.7 se obraz ve vrstvě vyplňuje funkcí FILL. Kotva obrázku je definována funkcí ANCHOR. Obě funkce se vyjadřují v souřadnicovém systému x-y, které jsou nastaveny tak, že v levém horním rohu je [x y]=[0 0] a v pravém dolním [x y]=[1 1], přičemž osa x je položená horizontálně a osa y vertikálně. Pomocí funkce FILL je definováno v jakém místě bude umístěná kotva obrázku vůči souřadnicím monitoru. Funkcí ANCHOR

je definováno v jakém místě bude obrázek ukotven. Kotva se nastavuje podle souřadnic pevně vázaných na obrázek. To znamená, že pokud se otočí obrázek, otočí se i jeho souřadnice. Pokud tedy například chci, aby byl umístěn uprostřed obrazu, byl přesně uprostřed také ukotven, a tudíž osa otáčení byla uprostřed, je třeba nastavit FILL na [x y]=[0.5 0.5] a ANCHOR taktéž na [x y]=[0.5 0.5]. Příkazem FILL se dále nastavuje škálování obrazu. Hodnoty x_scale a y_scale definují poměr původního obrazu ku zobrazovanému. Z toho vyplývá, že pokud je obraz v nezměněné velikosti, hodnoty jsou rovny 1 a hodnotami blížícími se k nule se obraz procentuálně zmenšuje.

Příkaz pro výplň plochy vrstvou vypadá například takto:

>> MIXER 1-10 FILL 0.25 0.25 0.5 0.5 25 easeinsine

První dvě hodnoty po názvu funkce FILL definují polohu kotvy obrázku na souřadnicích monitoru, druhé dvě hodnoty nastavují škálování obrazu. Vždy je první osa x, poté osa y. Poslední dvě hodnoty se týkají animace přechodu z původních hodnot na nově nastavené. Číselná hodnota, zde 25, vyjadřuje rychlost přechodu vztaženému k počtu framů. Poslední hodnotou je typ animace přechodu, které jsou podrobněji popsané níže.

Příkaz pro nastavení kotvy vrstvy:

>> MIXER 1-10 ANCHOR 0.5 0.5

V tomto případě by byla kotva nastavena na přesný střed obrázku.

Rotation Pokud jsou nastavené hodnoty FILL a ANCHOR, je možné obrázek otočit o daný úhel. Slouží k tomu funkce ROTATION, která otočí obrázek (vrstvu) ve směru hodinových ručiček o zadaný úhel ve stupních s osou otáčení v kotvě.

Příkaz pro otočení vrstvy:

>> MIXER 1-10 ROTATION 45

Tento příkaz otočí obrázek o 45 stupňů doprava.

Perspective V nové verzi číslo 2.0.7. přibyla také nová funkce, a to funkce pro nastavení perspektivy vrstvy - PERSPECTIVE. Tu lze nastavit pomocí množiny osmi čísel v intervalu od nuly do jedné, kdy každá dvojice čísel vyjadřuje polohu rohu vrstvy v souřadnicích monitoru.

Příkaz pro nastavení perspertivy:

>> MIXER 1-10 PERSPECTIVE 0.4 0.4 0.6 0.4 1 1 0 1

Pro každý roh je osa x vyjádřená prvním číslem a osa y druhým.

Obrázek 2.2: Příklad použití funkce Perspective

Dvojice popisující rohy jsou seřazeny podle směru hodinových ručiček, tedy: horní levý roh (top_left), horní pravý roh (top_right), spodní pravý roh (bottom_right) a spodní levý roh (bottom_left).

Výsledek může ve spojení více vrstev vypadat jako na obrázku 2.2

Clip Jako poslední funkci transformace vrstvy lze zařadit také funkci CLIP, která umožňuje ořezávat vrstvu podle předem definovaných souřadnicových hodnot.

Tuto funkci asi bude nejsnazší rovnou vysvětlit na příkladu příkazu:

>> MIXER 1-10 CLIP 0.25 0.25 0.5 0.5 25 easeinsine

První dvě hodnoty pro osu x a osu y určují, na jakém bodě souřadnic monitoru bude leže levý horní roh (nula) vyříznutého obrazu. Druhé dvě hodnoty definují velikost ořezávacího rámečku. Pro animovaný přechod lze stejně jako u FILL definovat dobu trvání přechodu a jeho typ.

Chroma Filtr CHROMA dokáže vyklíčovat danou barvu v dané vrstvě. Používá se převážně, pokud je obraz natočen nebo nafocen na klíčovacím pozadí.

Příkaz k použití vypadá takto:

>> MIXER 1-10 CHROMA green 0.10 0.20

Příkazem se nastavuje jak barva klíčování, tak i spodní a horní práh barvy.

Obrázek 2.3: Příklad použití filtru Blend Mode

Blend Mode Pomocí filtru BLEND MODE můžeme nastavit režim mísení vrstev. Tato funkce vychází z editorů, jako například Adobe Photoshop, které umí za použití určitého algoritmu smíchat dvě vrstvy a dodat jim tím nějaký efekt. Příkaz pro nastavení této funkce vypadá následovně:

>> MIXER 1-10 BLEND Phoenix

Výsledné spojení modrého pozadí a fotky červené chryzantémy může při použití módu Phoenix jako na obrázku číslo 2.3

Opacity U vrstev lze také samozřejmě nastavit jejich průhlednost. K tomu slouží funkce OPACITY. Přechod této funkce lze animovat pomocí tweener přechodů – viz níže.

Příkaz pro nastavení průhlednosti:

>> MIXER 1-10 OPACITY 0.5 25 easeinsine Hodnotu průhlednosti lze nastavit v intervalu 0 až 1, kdy 0 znamená absolutní průhlednost a 1 pro neprůhlednou vrstvu.

Brightness Brightness, jas, lze nastavit pomocí příkazu:

>> MIXER 1-10 BRIGHTNESS 0.5 25 easeinsine

Lze ho nastavovat v hodnotách 0 až 1, přičemž 0 znamená absolutní černou a 1 originální jas. Jas můžeme také zvyšovat oproti originálnímu a to hodnotami

vyššími než 1. Horní hranice není definována. Přechod lze, stejně jako většinu filtrů, animovat.

Saturation Sytost barev a saturace se nastavují pomocí příkazu SATURATION. Hodnoty saturace mohou být ve stejném intervalu jako hodnoty jasu. To znamená, že 0 je pro nulovou sytost barev – pouhé odstíny šedi a 1 pro originální obraz. Hodnotami vyššími než jedna můžeme sytost zvyšovat.

Příkaz vypadá takto:

>> MIXER 1-10 BRIGHTNESS 0.5 25 easeinsine

Jak je patrné z příkazu, přechod na tento filtr lze animovat.

Contrast Příkaz CONTRAST definuje kontrast vrstvy. Hodnota 0 znamená nulový kontrast, šedou, hodnota 1 originál a hodnoty vyšší než 1 kontrast dále zvyšují.

Kontrast a animaci jeho přechodu lze nastavit tímto příkazem:

>> MIXER 1-10 CONTRAST 0.5 25 easeinsine

Levels Pomocí tohoto příkazu lze nastavovat převodní křivku mezi vstupem a výstupem. Něco podobného jako se používá u foto-editorů.

Hodnoty lze nastavit následujícím příkazem:

>> MIXER 1-10 LEVELS 0.1 0.1 1.0 0.9 0.9 25 easeinsine

Pětice hodnot vyjadřuje min_input, max_input, gamma, min_output, max_output. Všechny hodnoty lze nastavovat v intervalu 0 až 1.

Volume Použít lze také filtry na základní úpravu hlasitosti zvuku. Příkaz VOLUME nastavuje hlasitost jedné, předem definované, vrstvy.

Definuje se takto:

>> MIXER 1-10 VOLUME 0.5 25 easeinsine

Stejně jako u filtrů pro úpravu obrazu lze i přechod na tento filtr animovat pomocí tweener přechodů.

Master Volume Pro úpravy hlasitosti celého kanálu lze použít příkaz:

>> MIXER 1 MASTERVOLUME 0.5

Hodnoty hlasitosti lze nastavovat v intervalu 0 až 1, kde 0 znamená ztlumení a 1 originální hlasitost.

Mixer Clear Pro vymazání veškerých transformací lze použít příkaz MIXER CLEAR.

Lze ho použít buď pro vymazání transformací celého kanálu:

>> MIXER 1 CLEAR

Nebo pouze pro danou vrstvu:

>> MIXER 1-10 CLEAR

Grid Tento příkaz vytvoří mřížku vrstev srovnaných podle vzestupně podle indexů vrstev. Počet dílů mřížky se nastavuje celočíselnou hodnotou, kdy například hodnota 2 znamená mřížku 2x2, hodnota 3 3x3 a tak dále.

>> MIXER 1 GRID 3

Přechody Tweener Přechody Tweener[5] jsou široce používané přechody postavené na jazyce ActionScript 3. V modulu Mixer je lze využít v rámci většiny funkcí.

Consumers

Screen Consumers První a asi nejvyužívanější modul je Screen Consumer, který dokáže zobrazit hlavní výstup na monitoru. Lze nastavit jak poměr stran, vyplnění obrazu přes celý monitor nebo ho umístit do klasického rámečku okna a další možnosti upřesňující výsledný vzhled, tak například na jakém zařízení (monitoru) se náhled spustí.

BlueFish Consumers Tento modul umí přehrát výstup na BlueFish video kartě s plnou SDI podporou pro všechny SD a HD rozlišení, snímkovací frekvence včetně odděleného alfa kanálu a audia.

DeckLink Consumers Modul pro řízení výstupu před DeckLink video karty na SDI nebo HDMI.

Disc Consumers Díky tomuto modulu lze, již bylo zmíněno v podkapitole 2.2, ukládat výstupní videodata na disk počítače.

Příkaz pro spuštění nahrávání vypadá například takto:

>> ADD 1 FILE myfile.mov

[5] http://hosted.zeh.com.br/tweener/docs/en-us/misc/transitions.html

A pro zastavení nahrávání:
>> REMOVE 1 FILE

Stream pomocí FFmpeg FFpmeg je volně šiřitelný program pro rychlou, ale zato velmi propracovanou konverzi audia a videa. CasparCG má v sobě tento program implementován, proto lze veškerá nastavení provádět pomocí přístupu přes telnet. Tato nastavení jsou prováděna pomocí argumentů, které parametrizují jednotlivé moduly programu a nastavují jim požadovanou hodnotu. Operace s jednotlivými moduly je prováděna přesně podle zápisu, z toho důvodu je potřeba při zápisu dodržovat i pořadí jednotlivých argumentů. Samotné nastavení hodnot argumentů je potřeba přizpůsobit možnostem streamovacího počítače. Široké spektrum všech argumentů a způsobů použití je uvedeno na stránkách produktu. Před použitím tohoto modulu je dobré nastudovat alespoň základní způsoby použití.[4]

2.2.3 Konfigurace serveru

Veškerá konfigurace probíhá pomocí XML souboru caspar.config, který lze jednoduše editovat v jakémkoliv textovém editoru. Ukázkou plnohodnotného konfiguračního souboru je algoritmus číslo 2.1.

Celá konfigurace je psaná pomocí jazyka XML, proto jsou všechny větve kódu rozděleny pomocí tzv. tagů. Jako první se pomocí tagů <paths> a </paths> definuje část kódu pro nastavení umístění kořenových složek pro soubory uložené na disku. Dalším v pořadí je nastavení kanálu mezi tagy <channels></channels>. Kanálů může být i více, pouze je třeba nastavit jim správné indexy. Každý kanál pak může mít rozlišené nastavení. Nejprve se nastavuje rozlišení výstupu. CasparCG Server podporuje jak PAL a NTSC formát, tak i v podstatě všechny kombinace rozlišení a snímků za sekundu, až od verze číslo 2.0.4 po rozlišení 4K při snímkování až 30 snímků progresivně. Zápis takového formátu v konfiguraci pak vypadá takto: 2160p3000. Následuje nastavení výstupu. Zde lze nastavit veškeré typy výstupu, jako například náhled na monitor, výstup pomocí grafické karty DeckLink, BlueFish, ukládání do souboru, či stream. Výstup lze také směrovat na další hardwarovou video-střižnu jako například Blackmagic Design ATEM či Sony a k těmto účelům lze kromě klasického barevného výstupu nakonfigurovat také paralelní výstupní alfa-kanál. Následuje nastavení zvuku, případně možnost nastavení zpoždění výstupu. V sekci controllers se nastavuje už jen port, na kterém server běží a protokol používaný pro komunikaci. Více informací a možností

Algoritmus 2.1 Ukázka konfiguračního souboru caspar.config

```
<?xml version="1.0" encoding="utf-8"?>
 <configuration>
  <paths>
   <media-path>media\</media-path>
   <log-path>log\</log-path>
   <data-path>data\</data-path>
   <template-path>templates\</template-path>
   <thumbnails-path>thumbnails\</thumbnails-path>
  </paths>
  <channels>
   <channel>
    <video-mode>pal</video-mode>
    <consumers>
     <screen>
     </screen>
     <system-audio></system-audio>
    </consumers>
   </channel>
  </channels>
  <controllers>
   <tcp>
    <port>5250</port>
    <protocol>AMCP</protocol>
   </tcp>
  </controllers>
</configuration>
```

konfigurace je jak na webu autora[6], tak zakomentované v konfiguračním souboru.

2.3 Klient

Pro jednoduché úkony lze používat klienta, který je dostupný na stránkách autora v sekci download[7]. Jedná se však o klienta především demonstrujícího všechny funkce, kterými server disponuje. Pro náročnější a specifické úkony je dobré naprogramovat si klienta svého. Většina klientů je tedy navržena pouze pro daný účel – co nejjednodušší posílání instrukcí serveru, například při fotbalovém zápase, kde lze jednoduše naprogramovat například změnu skóre stisknutím jediného tlačítka. Pro jiné účely lze také naprogramovat složitějšího klienta, například za účelem tvorby playlistů. V klientovi se jednoduše vytvoří playlist se všemi videi, animacemi či šablonami, které se pak podle předem daného postupu spustí a vše se tím zautomatizuje.

2.3.1 CasparCG Client 2.0.7

Klient pracuje s kořenovými složkami serveru (podle toho, jak je má server nakonfigurované) pro jednotlivé typy vstupů – Audio, Image, Template a Video. Zdrojové soubory se z daných složek přetažením načtou a lze s nimi dále pracovat. V první řadě je potřeba nastavit každému vstupu číslo vrstvy, na které bude v serveru umístěn. Defaultně mají videa nastavené číslo vrstvy 10, šablony 20 a zvuk 30. Pokud je ovšem více vstupů stejného typu, je potřeba tyto soubory rozvrstvit do více vrstev podle toho jak se mají překrývat. Pomocí funkcí v záložce Mixer lze dané vrstvy dále transformovat a upravovat, podle možností serveru, které jsou vypsány v podkapitole 2.2.2, odstavci MIXER. Pro účely této práce byl naprogramován vlastní klient v programovacím jazyku Python. O tomto klientovi nazvanému BPstream pojednává čtvrtá kapitola.

[6] http://casparcg.com/wiki/CasparCG_Server#CasparCG_Server_Configuration
[7] http://www.casparcg.com/download

2.4 Tvorba flashové šablony

2.4.1 Generátor šablon

Pro vytvoření flashové šablony se nejčastěji používá Adobe Flash Professional a Adobe After Effect. Pro tyto programy existuje rozšíření, takzvané Generate Template, které lze stáhnout ze stránek autora a snadno do programu nainstalovat. V případě Adobe Flash Professional stačí stažené rozšíření spustit a samo se naimportuje do programu. Při použití rozšíření pak lze vytvořenou šablonu pomocí generátoru vyexportovat ve formátu .ft, formátu kompatibilním s CasparCG serverem. Pomocí tohoto rozšíření lze přímo propojit program pro tvorbu šablon se serverem a rovnou funkci šablony testovat. Šablonu lze také jakkoliv naprogramovat pomocí jazyka Action Script.

KAPITOLA

TŘETÍ

TWITTER API

3.1 Twitter

Twitter je známá, stále se rozšiřující sociální síť, určená především pro zasílání krátkých textových zpráv. Příspěvky zvané tweety mohou obsahovat buď takzvaný hashtag, klíčové slovo označené znakem křížku '#', nebo odkazování na další uživatele Twitteru pomocí znaku zavináče '@'. Díky těmto klíčovým slovům může být tweet adresován dané události nebo osobě, stejně tak lze tweety podle těchto klíčových slov filtrovat a vyhledávat. Pomocí twitteru uživatelé mohou sdílet svou aktuální náladu, komentovat dění ve svém okolí a podobně.[5]

3.1.1 Twitter API

Twitter API (application programming interface) povoluje aplikaci sdílet data se zbytkem světa. API stojí na dotazu pomocí URL a strukturované odpovědi v datovém formátu nejčastěji XML nebo JSON, které lze snadno analyzovat a s daty dále pracovat.

Twitter API povoluje tři typy HTTP dotazů. Prvním typem je **GET**, metoda, která je využívána k získávání informace na jiném serveru pomocí URL dotazu. Informace může být generována pomocí například PHP, kdy se podle tvaru URL adresy vygeneruje obsah v požadovaném formátu.

Například adresou https://twitter.com/favorites.xml budou vypsány všechny oblíbené tweety přihlášeného uživatele.

Dalším typem, metodou **POST**, je získáván stejný výsledek, jako metodou GET,

avšak jiným způsobem. Metodou POST na rozdíl od metody GET není zadáván požadavek pomocí specifického tvaru URL adresy, která může být navíc pouze omezeně dlouhá, ale je zasílán skrytě. Díky tomu lze pokládat více dotazů, zatímco URL adresa po položení dotazu zůstává stejná.

Pomocí adresy https://twitter.com/statuses/update.xml a k ní příslušných hodnot lze touto metodou aktualizovat status uživatele - to znamená vytvořit nový tweet.

Za zmínku stojí ještě poslední typ dotazu, a to **DELETE**. Účelem tohoto HTTP volání je instruovat server k odstranění daného záznamu. Tato metoda však nemůže garantovat, že daný záznam opravdu odstraněn byl.

Twitter aktuálně používá čtyři typy datových formátů. Prvním typem je asi nejznámější datový formát a to **XML** (Extensible Markup Language). Tento formát je oddělován pomocí takzvaných tagů, kterými je strukturován pomocí hierarchie typu strom.

Datový formát **JSON** (JavaScript Object Notation) je jazykově nezávislý textový formát používaný primárně pro AJAX aplikace. Stejně jako u XML jsou data uložena v jasně daném strukturovaném formátu, ačkoli JSON se jeví jako jednodušší než XML.

Typ **RSS** (Really Simple Syndication) je specifická forma XML, která vrací data strukturovaná pomocí standardizovaných tagů. Díky nim může být tento typ přečten předvídatelným způsobem. Nejčastěji je tento datový formát využíván u rozšíření různých blogů, stránek s aktuálními zprávami a často aktualizovanými informacemi anebo právě službami typu Twitter.

Alternativou k RSS je datový formát **Atom**, který byl vytvořen kvůli odstranění nutnosti podpory starších protokolů. Atom oproti RSS používá jiný datový typ, čímž se snáze přizpůsobí mezinárodní podpoře.[6]

3.1.2 Autentizace

Většina API dotazů vyžaduje autentizaci pomocí uživatelského jména a hesla. Tato autentizace je nutná ze dvou důvodů. Za prvé, že některá data jsou určena pouze pro ověřeného uživatele a za druhé, autentizace je nejspolehlivější způsob, jak usnadnit omezení přístupu k API. Tato omezení jsou nutná pro ochranu před zneužitím dat třetími stranami. Od roku 2009 je zavedeno ověřování pomocí OAuth a následně OAuth2.

Každá aplikace musí být registrována na Twitteru[1]. Po registraci Twitter vrátí přístupové klíče, tokeny, potřebné pro autentizaci dotazu aplikace.

3.1.3 Tweet

Informace vrácené Twitterem mohou obsahovat jeden nebo více paramterů. Většina z nich je volitelná, ale některé jsou fixně dané. Zároveň ne všechny parametry lze použít u všech metod.

Twitter API používá pro všechny hodnoty parametrů UTF-8 kódování, to znamená, že také veškeré posílané hodnoty musí být kódované. Mezi nejdůležitější parametry, které tweet obsahuje, patří: *id/user_a* a *user_b/id*, ve kterých je obsaženo ID uživatele, oproti tomu parametr *user/screen_name* vrací přímo uživatelské jméno uživatele. Parametr *text* obsahuje tu nejdůležitější část tweetu, a to samotný text, který je limitován na 140 znaků. Parametr *location* obsahuje údaj o poloze uživatele a parametr *device* údaj o zařízení, kterým byl tweet poslán.

Jeden tweet obsahuje ještě mnoho dalších parametrů, které jsou všechny vypsány na webové stránce Twitteru pro vývojáře[2]. Pro účely této práce byly použity pouze parametry *text* a *screen_name.*

[1] https://apps.twitter.com/
[2] https://dev.twitter.com/overview/api/tweets

KAPITOLA

ČTVRTÁ

KLIENT BPSTREAM

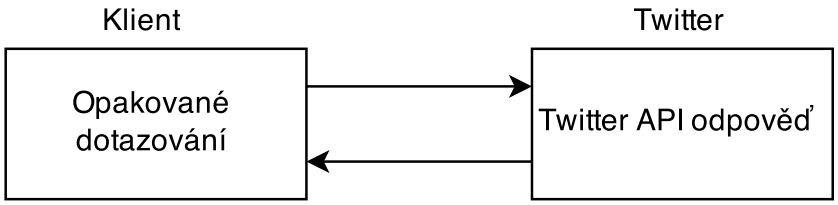
4.1 Python

Na oficiální stránkách je Python definován jako objektově orientovaný programovací jazyk, často srovnáván s jazyky typu C#, Visual Basic nebo Java. Jazyk Python byl použit především pro svou jednoduchost a velmi dobrou podporu ohledně telnetových knihoven či knihoven pro práci s Twitterem. Za zmínku také stojí program PyCharm 4.0.6 pro programování v tomto jazyce, nabízející velmi pěkné a přehledné uživatelské prostředí a zároveň široké možnosti pro usnadnění práce.[7]

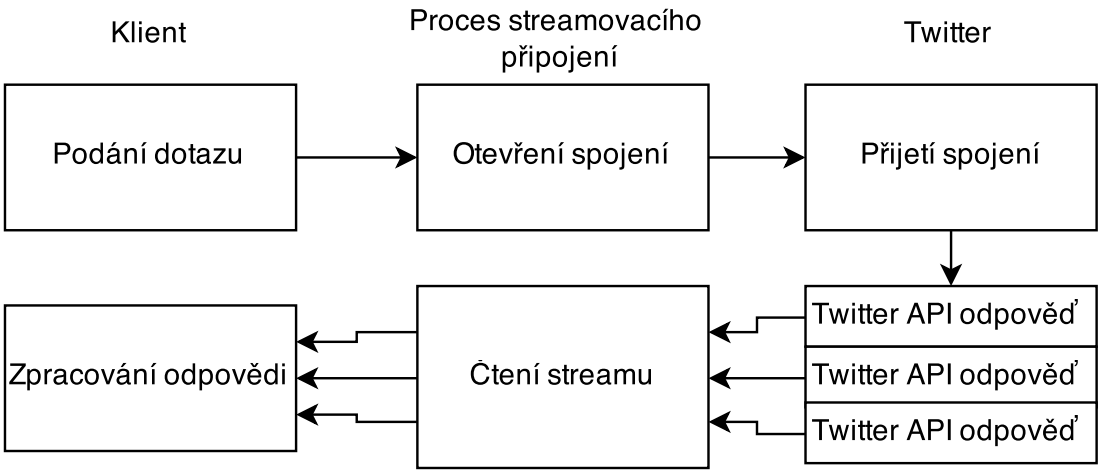
4.2 Twython

Na vývojářských stránkách[1] Twitteru je uveden seznam knihoven, které nabízí práci s API, avšak každá je vhodná k jinému účelu. Pro tuto práci se jevila jako nejvhodnější knihovna zvaná Twython díky možnosti využití způsobu komunikace zvaného *Streaming API.* Rozdíl mezi tímto způsobem připojení a klasickým typem komunikace zvaným *REST API* je ten, že zatímco při použití REST API je nutné se opakovaně dotazovat na server Twitteru, zda se neobjevil nový příspěvek, čímž se tento proces znatelně zpožďuje a mezi klientem a serverem tečou zbytečná data, Streaming API nabízí vytvoření komunikace server-server. V rámci klienta se vytvoří běžící server, na který poté pouze čte stream dat z Twitteru. Z toho

[1] https://dev.twitter.com/overview/api/twitter-libraries#python



Obrázek 4.1: Blokové schéma procesu Twitter REST API



Obrázek 4.2: Blokové schéma procesu Twitter Streaming API

vyplývá, že mezi klientem a serverem probíhá po vytvoření spojení komunikace pouze ve chvíli, kdy je přidán nový příspěvek. Tím je možné dosáhnout již zmíněné minimální latence. Oba procesy jsou znázorněny blokovými schématy číslo 4.1 a 4.2.[8]

4.3 Funkce klienta

4.3.1 AMCP protokol

Připojení k serveru

Server lze spustit na IP adrese 127.0.0.1, takzvaný localhost - spuštění na lokálním zažízení. Stejně tak jej ale lze spustit na jakémkoliv počítači s veřejnou IP adresou a pak k němu přistupovat vzdáleně. Port je defaultně nastaven na 5250, avšak číslo

portu lze v případě potřeby jednoduše změnit v konfiguračním souboru číslo 2.2.3. Pro připojení telnetového klienta je použita knihovna telnetlib.py, která součástí originálního balíku Python.

Dvě nejdůležitější funkce této knihovny jsou funkce *open()* a *close()*, které zajišťují připojení a odpojení k určitému serveru. Funkce open je pak nutné nastavit vstupní parametry *host* a *port* serveru.

Příkazy pro práci s daty šablon

Veškeré základní příkazy pro nahrávání, spouštění či ukončování videí jsou popsány v kapitole 2.2.1 CasparCG. Z toho důvodu jsou v této kapitole zmíněny pouze příkazy pro práci se šablonami.

Prvním stěžejním příkazem je **CGAdd**. Tento příkaz funguje podobně jako příkaz LoadBG, tedy že načte šablony na pozadí dané vrstvy, kde čekají na příkaz k přehrání. Na jedné vrstvě může být až 99 šablon, proto se zde definuje kromě čísla kanálu a čísla video-vrstvy i takzvaná flashová vrstva. Dalšími parametry příkazu jsou jméno šablony, parametr definující zda se šablona spustí rovnou s načtením, nebo až zavoláním nového příkazu (0 nebo 1), a na závěr samotná data obsahující text pro dynamická pole šablony. Tato data jsou posílána ve formátu XML, který je popsán v podkapitole 4.3.4.

Příkaz CGAdd může mít následující podobu:

>>CG 1-10 ADD 10 nazev_sablony 1 data_sablony

Pro případ zvolení u příkazu CGAdd možnosti pouze načíst šablonu na pozadí je určen příkaz **CGPlay**. Tento příkaz spouští (viditelně) určenou flashovou vrstvu.

Pro ukončení aktuálně přehrávané šablony slouží příkaz **CGStop**. Odstranění viditelné šablony ze specifické flashové vrstvy lze také provést příkazem **CGRemove**, avšak s tím rozdílem, že při použití příkazu stop se provede (pokud je tak v šabloně přednastavené) animace k ukončení šablony. Při použití CGRemove se šablona a její data okamžitě odstraní.

Celou video-vsrtvu včetně všech flashových vrstev v ní obsažených lze vyčistit pomocí příkazu **CGClear**.

Některé šablony jsou nastaveny tak, že se vždy v daném bodě pozastaví a čekají na příkaz, který jim umožní pokračovat v animaci do dalšího definovaného bodu. Tímto příkazem je přákaz **CGNext**.

Veškeré tyto příkazy mají podobný zápis:

>>CG 1-10 PLAY 10

>>CG 1-10 STOP 10
>>CG 1-10 REMOVE 10
>>CG 1-10 CLEAR
>>CG 1-10 NEXT 10

Většina šablon se využívá především kvůli možnosti použití dynamického textu, který lze v průběhu přehrávání šablony měnit, a to pomocí příkazu **CGUpdate**. Data jsou posílána ve stejném formátu, jako u CGAdd.

Příkaz pak bude mít následující tvar:
>>CG 1-10 UPDATE 10 data_sablony

Příkazy pro práci s modulem MIXER

V této práci je použito několik příkazů modulu MIXER. Veškeré tyto příkazy jsou popsány v odstavci 2.3.1, pojednávající o samotném programu.

4.3.2 BPplay

Po spuštění této sekce se systém zeptá, zda chcete navázat na předchozí hlasování - načíst již vyplněnou databázi, anebo začít s novým playlistem a vytvořit tedy databázi novou, prázdnou. Playlist je potřeba uložit v XML formátu a mít přesně danou strukturu, znázorněnou algoritmem číslo 4.1. Pokud je tedy zvoleno načtení nového playlistu, uživatel je vyzván k zadání celého názvu playlistu, to znamená například *playlist.xml*. Poté se vytvoří databáze, do které se vytvoří čtyři tabulky.

První a hlavní se jmenuje ***score***. Má devět sloupců obsahující informace jak o názvu, adrese nebo délce videa - informace na základě načtení playlistu, tak o aktuálním stavu hlasování. Sčítají se zde všechny hlasy pro a proti či počet přehrání videa. Sloupce *now* a *sequence* jsou čistě pro chod programu. Sloupec *now* je defaultně pro každé video nulový a hodnotu 1 má pouze to video, které je právě přehráváno. Program běží na více vláknech a je složité a trochu i riskantní komunikovat přímo mezi vlákny. Proto je zde využito externí databáze, do které mohou vlákna nezávisle na sobě zapisovat nebo z ní číst. Sloupec *sequence* popisuje v jaké sekvenci bylo video naposledy přehráno

Do tabulky ***comment*** se ukládají veškeré příspěvky, které klient vyfiltruje z Twitteru. Tabulka ***PlayNext*** slouží převážně k účelům programu. Zaznamenává hodnoty pro komunikaci mezi vlákny týkající se hlasování o následujícím videu. Pro

Algoritmus 4.1 ukázka playlistu ve formátu XML

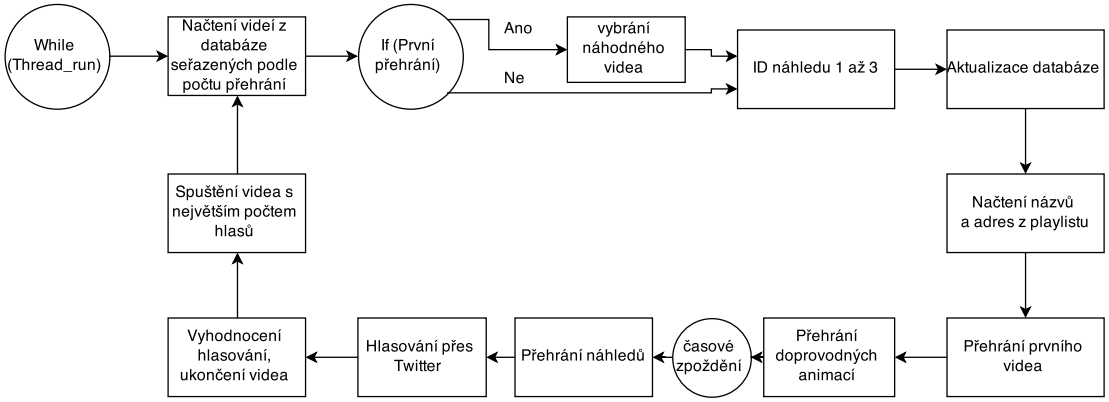
```
<?xml version="1.0"?>
<data>
  <video num="1">
    <name>Název videa</name>
    <address>Adresa</address>
    <length>délka v sekundách</length>
    <likes></likes>
    <dislikes></dislikes>
    <played></played>
  </video>
</data>
```

správný chod je potřeba znát číslo aktuální sekvence spojené s id videa. To zajišťuje tabulka ***sequence***.

V tuto chvíli, kdy je připravená databáze s aktuálním playlistem, se spustí logo a pozadí. To je uložené v kořenové složce pod jménem background.png. Lze ho jakkoliv měnit podle požadavků uživatele. Musí však jen splňovat poměr stran 16:9, minimální rozměr 1920:1080 a formát png, aby zůstala zachovaná stejná adresa souboru. Poté začíná přehrávání samotných sekvencí.

Sekvence

Sekvence začíná vynulováním sloupce *now* v tabulce *score* z předešlého přehrávání. Samotný cyklus popisuje blokové schéma číslo 4.3. Cyklus probíhá do té doby, dokud je proměnná *Thread_Run* pravda. V samotném cyklu se nejprve načtou videa uložená v databázi. Pomocí SQL příkazu *order by* jsou data seřazená podle nejmenšího počtu přehrání. Tím je zajištěno, že se videa budou přehrávat rovnoměrně často i přes to, že uživatelé mohou měnit pořadí přehrávání. Pokud se jedná o první sekvenci, vybere se pro náhodné video (při přehrávání dalších sekvencí v tuto chvíli již hlavní video běží). Dále jsou vytvořeny 3 náhledy pro hlasování o výběru dalšího videa. Ty jsou vybírány ze seřazeného seznamu, proto je stále zachováno, že se jedná o tři videa s nejmenším počtem dosavadních přehrání. V tento okamžik je aktualizována databáze. V tabulce *sequence* je vytvořen záznam o nové sekvenci a id aktuálního videa, v tabulce *score* je nastaveno *now* na hodnotu jedna a upraven počet přehrání. Pro spuštění prvního videa je třeba znát jeho adresu, název a délku, hodnoty, které jsou naimportovány z XML playlistu.



Obrázek 4.3: Blokové schéma cyklu sekvence

Během přehrávání videa jsou spouštěny animace pro zobrazení názvu videa, upomínka pro hlasování či načtení a přehrání náhledů dalších videí. Každá tato operace zapříčiňuje zpoždění odpočítávání času do konce videa. Z toho důvodu bylo využito počítadla času počítače, které zde slouží jako další vlákno nezávislé na běhu programu. Proto je při spuštění videa spočítán zároveň čas (v sekundách od 1.1.1970) v kolik sekund video skončí. Program si tento čas hlídá cyklem, a ve chvíli, kdy je aktuální čas rovný nebo větší než čas napočítaný, video se ukončí a přepne na nové.

Při spuštění náhledů následujících videí se zároveň vytvoří záznam v tabulce *PlayNext*, čímž se otevře možnost hlasování o následujícím videu. Hlasování je nastaveno na dobu 25% z celkového času aktuálního videa. Hlasovat lze přes twitter zadáním společného hashtagu pro celý program a přidáním vybrané volby. Výsledný tweet může mít například tuto podobu: #BPstream #2. Tímto se přidá hlas videu s náhledem číslo 2. Po skončení videa se do nejvyšší video-vrstvy přesune náhled videa s největším počtem hlasů, načtou se informace o aktuálním počtu like a dislike, video se roztáhne přes celou plochu a cyklus se opakuje.

4.3.3 BPtwitter

Vlákno BPtwitter řeší připojení k Twitteru, vyfiltrování určitého typu dat a následné uložení do databáze, případně vypsání výstupu CasparCG serveru. Celé toto vlákno je postaveno na veřejně dostupné knihovně Twython které je věnován odstavec 4.2.

Díky přístupovým kódům lze z Twitteru pomocí daného filtru stahovat vybrané příspěvky. Proces filtrování je popsán blokových schématem číslo 4.4. Pomocí so-

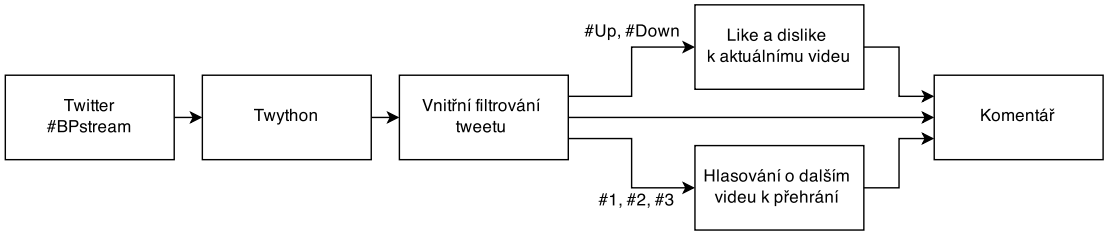
ciální sítě Twitter napíše uživatel příspěvek zvaný tweet, který obsahuje klíčové slovo zvané hashtag **#BPstream**. Twython vytvoří obousměrný komunikační kanál mezi Twitterem a klientem, a ve chvíli, kdy kdokoliv na světě přidá twitter s daným hashtagem, Twitter ho klientovi pošle. Knihoven pro práci s Twitter API a podobným stahováním tweetů je mnoho, avšak právě Twython dokáže příspěvky stahovat téměř v okamžiku, kdy byly přidány uživatelem. To výrazně snižuje zpoždění mezi akcí uživatele a viditelnou odezvou programu - například vypsáním tweetu ve výstupu programu.

Formátu a obsahu těla staženého tweetu je věnována podkapitola 3.1.3. Pro tuto práci je použit pouze samotný text tweetu a login jeho autora. V tomto ohledu je možné tuto práci dále rozvíjet o zobrazovaní obrázků obsažených v tweetu, státu, odkud byl příspěvek poslán a mnoho dalších informací. Avšak pro základní fungování tohoto klienta postačí pouze dvě zmíněné buňky pole.

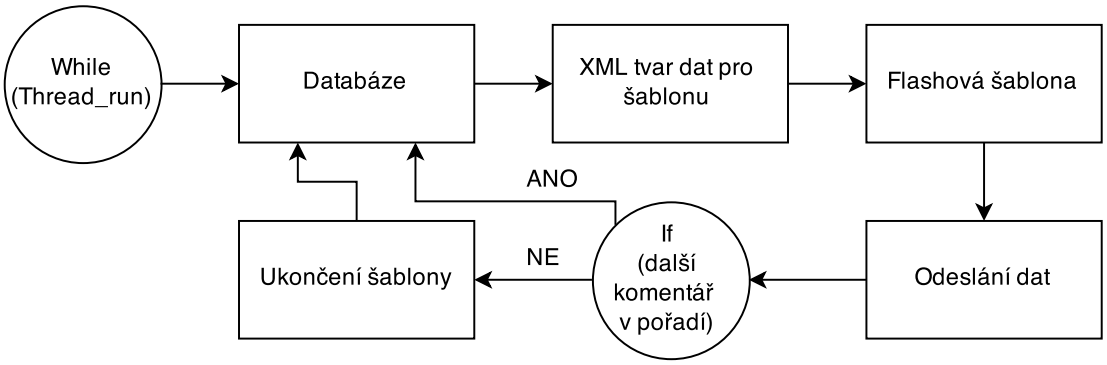
Výsledný stream, který vidí uživatel, navádí k posílání třech typů příspěvků. Prvním typem, základním, je hlasování o aktuálně přehrávaném videu. Pro jednoduchost klíčového slova pro tento účel byly použity hashtagy **#UP** pro vyjádření spokojenosti, v kódu se označuje jako like, a **#DOWN** pro nespokojenost, tzv. dislike, s aktuálním videem. V případě výskytu tohoto typu hashtagu se aktualizuje záznam k danému videu v tabulce *score*. Poté se aktualizuje flashová šablona zobrazující aktuální skóre videa.

Další možností hlasování přes Twitter je hlasování o následujícím videu. Před ukončením aktuálního videa jsou na určitý čas zobrazeny náhledy tří videí, z jichž jedno bude přehráno v následující sekvenci. Uživatel má možnost hlasovat o tom, jaké video to bude. Tvar hashtagu, který je zároveň u daného náhledu zobrazen, je jeho pořadové číslo ve formátu **#1**, **#2** nebo **#3**. Pokud tweet obsahuje tento hashtag, je v tabulce *PlayNext* v příslušné buňce záznamu patřícího k aktuální sekvenci k původní hodnotě přičteno +1. Po ukončení aktuálně přehrávaného videa se spustí video s největším počtem těchto hlasů.

Tweety, které nesplňovaly žádnou z podmínek hlasování, jsou brány jako komentář a vypíšou se pomocí příslušné flashové šablony do výstupu programu. Stejně tak se zachová veškeré hlasování, které se však ve výstupu zobrazí pouze na poloviční čas.



Obrázek 4.4: Blokové schéma popisující filtrování tweetu



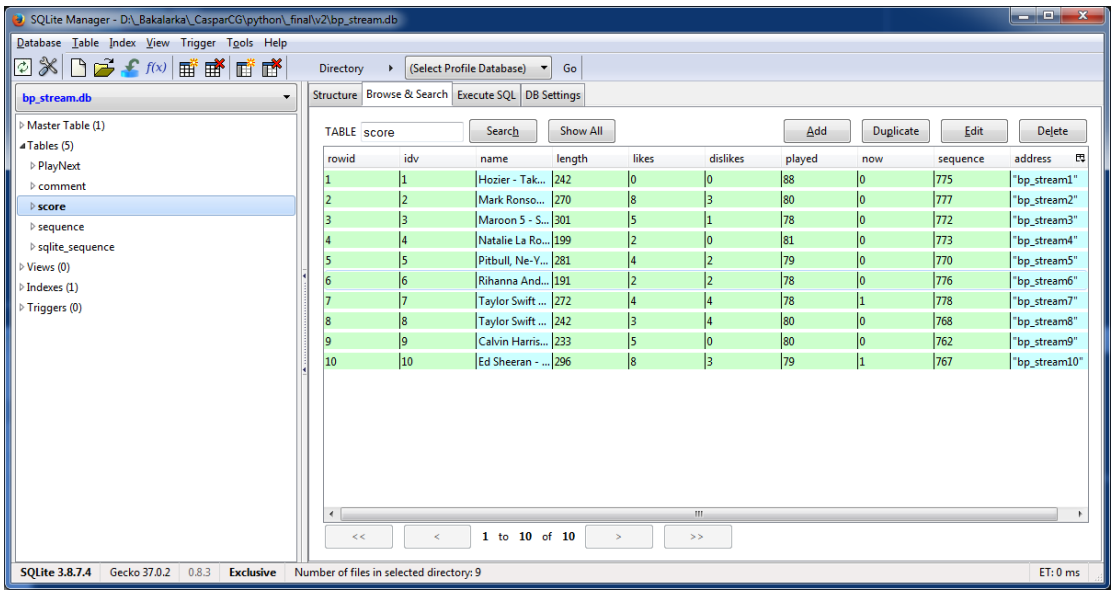
Obrázek 4.5: Blokové schéma cyklu pro vypisování tweetů

4.3.4 BPwriter

Toto vlákno zajišťuje vypisování veškerých komentářů do výstupu programu. Pro toto vlákno je stěžejní tabulka ***comment***, do které se ukládají veškeré komentáře. Tato tabulka obsahuje mimo jiné sloupec zvaný read. Každý tweet má defaultně nastavenou hodnotu read=1. Ve chvíli, kdy je tweet zobrazen na výstupu programu, změní se tato hodnota na hodnotu 0.

Cyklus, zobrazený na schématu číslo 4.5, vyhledává ve zmíněné tabulce řádek s hodnotou read=1. Samotný text tweetu je do šablony programu posílán ve formátu XML, kde tag <templateData> ohraničuje celá data, v něm vnořený tag <componentaData id="f0"> specifikuje pole, v tomto případě pole f0, a samotný text je pak vnořen pomocí tagu <data value="text tweetu">.

Do nově spuštěné flashové šablony je text posílán pomocí příkazu CGUpdate (4.3.1). V případě výskytu dalšího nezobrazeného tweetu šablona zůstane otevřená a pouze se aktualizuje její obsah. V případě vyčerpání nepřečtených tweetů se zvýší frekvence kontroly tabulky *comment* do chvíle, kdy je přidán vláknem BPtwitter nový komentář označený hodnotou read=1.



Obrázek 4.6: SQLite Manager

4.3.5 SQLite Manager

Pro práci s databází se nejlépe osvědčil velice jednoduchý program SQLite Manager (4.6). Jedná se o doplněk prohlížeče Mozilla Firefox, tudíž i samotná instalace je velice snadná. Pomocí tohoto doplňku lze pak kontrolovat správnou funkci aplikace a tok statistiky přehrávaných sekvencí.

KAPITOLA

PÁTÁ

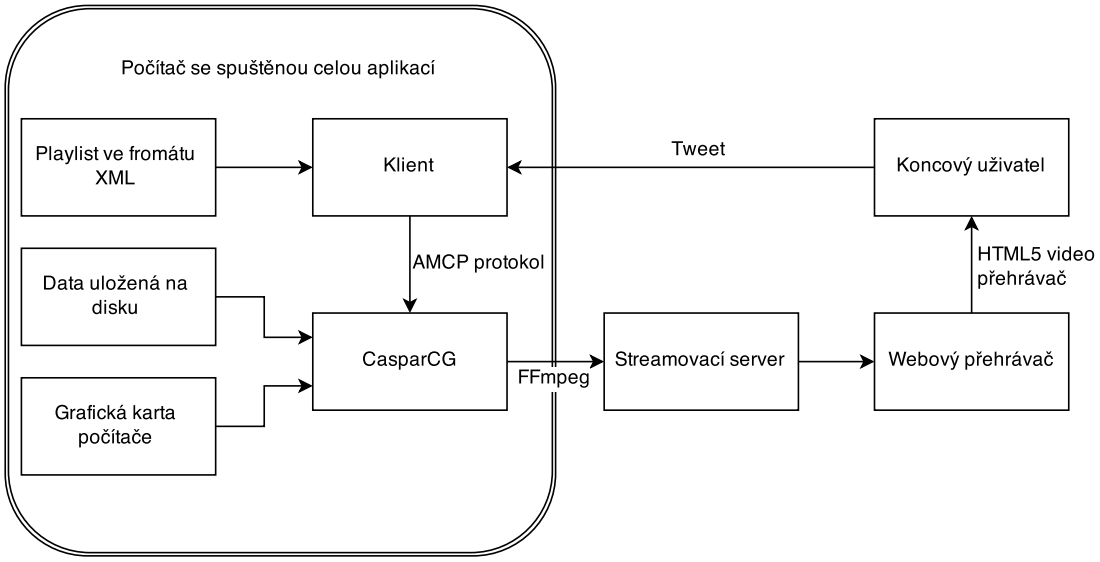
ZAPOJENÍ SYSTÉMU

5.1 Implementování aplikace do systému

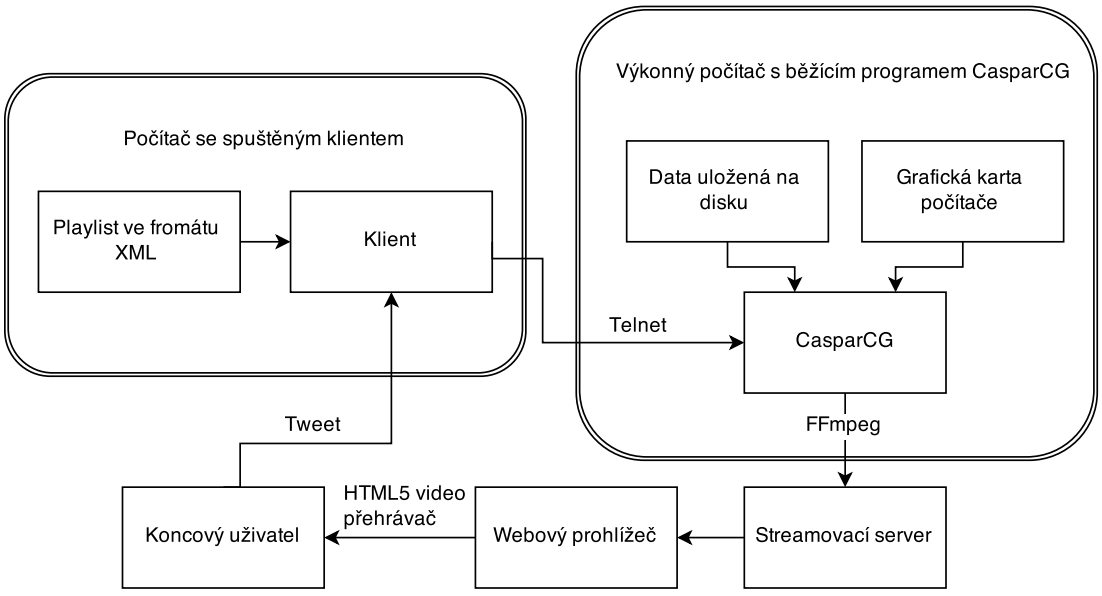
Implementování aplikace je zobrazeno na blokových schématech a popisuje zapojení od zdrojových souborů po konečného uživatele.

Celý proces začíná tím, že klient nahraje do své databáze playlist, popsaný v kapitole 4.3.2, který obsahuje seznam všech videí, včetně jejich adres a délky. Pomocí telnetu je pak přistupováno k programu CasparCG. Tomu jsou pomocí AMCP protokolu zasílány příkazy, jimiž jsou spouštěna jednotlivá videa zadaná v playlistu. Těmi mohou být jak videa uložená v počítači, tak například video vstupy grafické karty. Klient zároveň řídí spouštění veškerých animací a flashových šablon. Všechny tyto operace provádí na základě vyhodnocování informací z příchozích tweetů zadávaných konečnými uživateli. Jednoduché zapojení tohoto systému popisuje schéma číslo 5.1. To lze využít pro testování při používání nízkého rozlišení videí a streamu.

Programu CasparCG je pomocí FFmpeg modulu nakonfigurován výstup jako stream na danou adresu. Tento výstup musí být kódován do formátu X.264, volně šiřitelné alternativy k licencovanému formátu H.264. Tento formát je vyžadován většinou webových prohlížečů a není-li video v tomto formátu kódované, nelze ho jimi přehrát. V dnešní době je trendem ukládat či streamovat videa na internet v rozlišení nejlépe fullHD. Kódování tohoto rozlišení vyžadující datový tok kolem 4 až 5 Mbps je výpočetně velmi náročné. Vzhledem k tomu, že již samotný program CasparCG, který musí pracovat v případě této aplikace se čtyřmi fullHD videi a několika dalšími animacemi zároveň, je velmi náročný na výkon, je po při-



Obrázek 5.1: Blokové schéma zapojení s jedním počítačem



Obrázek 5.2: Blokové schéma zapojení se dvěma počítači

dání tohoto streamu náročnost o to zvýšena. Z tohoto důvodu bylo pro testování aplikace použito zapojení se dvěma počítači demonstrované blokovým schématem číslo 5.2. Díky tomuto zapojení lze pro běh aplikace použít méně náročný počítač a veškeré náročné operace tak přenést na výkonný počítač, na kterém je spuštěn CasparCG Server. CasparCG je v podstatě běžící server, na který lze přistupovat pomocí telnetu, a proto je možné posílat instrukce tomuto serveru (má-li veřejnou IP adresu) v podstatě odkudkoliv. Důležité pak je jen, aby měl server ve své kořenové složce nahraná všechna potřebná data, jako jsou videa, šablony a podobně. Pokud se jedná o opravdu velmi výkonný počítač, může sloužit zároveň i jako streamovací server.

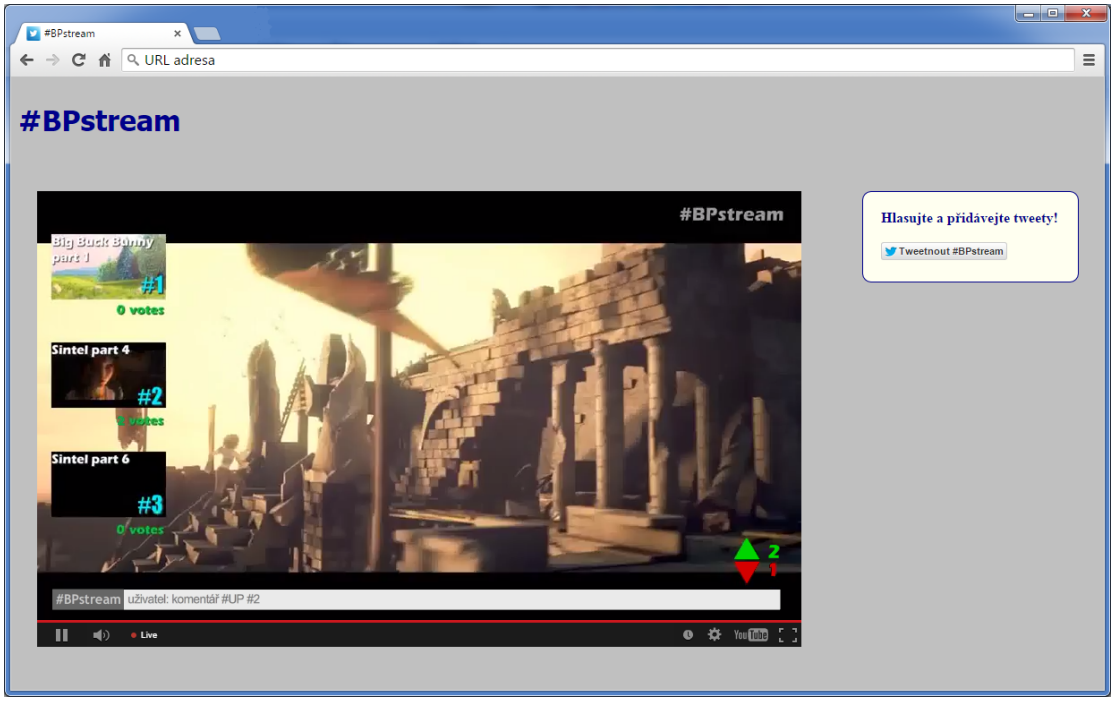
Jak již bylo zmíněno, CasparCG dokáže překódovat video do X.264 formátu a poslat ho na danou adresu a port. Jedná se však o unicast, to znamená, že stream může odchytávat pouze jedno zařízení. Pro více uživatelských přístupů najednou je třeba oddělený stream pro každého uživatele. Tento přístup je pak velmi náročný jak na výpočetní techniku streamovacího počítače, tak na rychlost připojení k síti. Z tohoto důvodu je dobré signál adresovat na streamovací server, který data přijme a dále je vysílá jako multicast.

Odchytávat stream lze pomocí k tomu určeného programu, například VLC Playeru, anebo pomocí webového prohlížeče. Tato možnost je velice usnadněná s příchodem HTML5, která téměř plně odstraňuje nutnost použití flashových přehrávačů pro přehrání streamovaného videa. V tuto chvíli již i gigant mezi video servery, Youtube, přešel na HTML5 video přehrávač. Na obrázku číslo 5.3 je ukázka jednoduchého uživatelského prostředí pro sledování streamu a jednoduchou možnost hlasování.

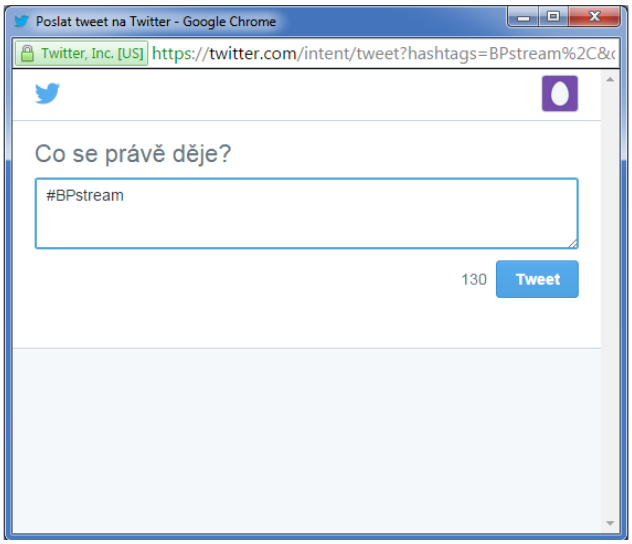
Koncový uživatel má tak možnost zároveň sledovat živé vysílání a zároveň jednoduše hlasovat pomocí tweetů, který lze v rámci této stránky jednoduše napsat. Po kliknutí na tlačítko se otevře nové okno zobrazené na obrázku číslo 5.4, kde je již předepsán potřebný hashtag. Dále může uživatel napsat libovolný komentář či hlasovat. Prostřednictvím tohoto klienta je zajištěno i přihlašování a identifikace uživatele.

5.1.1 Testování streamu pomocí programu VLC Player

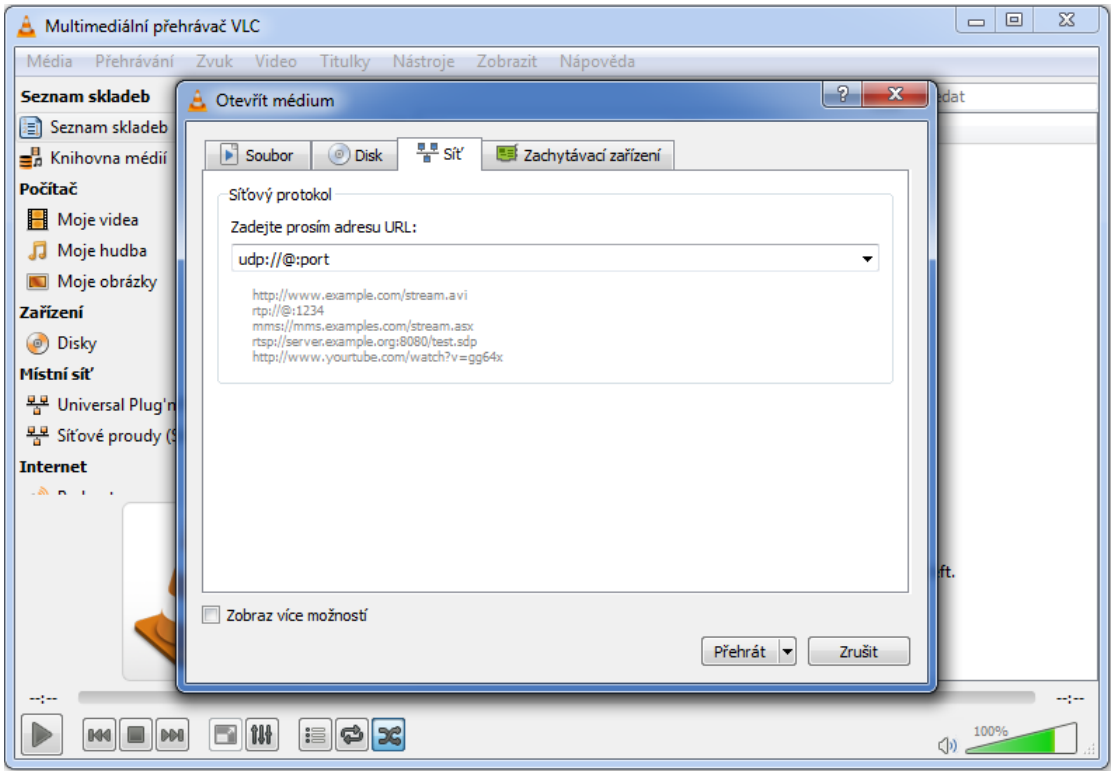
Pro testování je možné nastavit stream jako broadcast do určitého okruhu sítě. Přijímat ho však může pouze jedno připojené zařízení. Toto testování lze provést pomocí programu VLC Player, který je schopen odchytávat síťový proud na dané



Obrázek 5.3: Webová stránka s HTML5 video přehrávačem a možností přidání Twitteru



Obrázek 5.4: Twitter - rozhraní pro poslání tweetu z vlastních webových stránek



Obrázek 5.5: Nastavení VLC Playeru pro odchytávání streamu

adrese a portu. Příkaz pro nastavení streamu má například tuto podobu:

>>ADD 1 STREAM "udp://adresa_broadcastu:port" -f mpegts -acodec libvo_aacenc -ac 2 -ar 48000 -ab 192k -vcodec libx264 -preset veryfast -threads 2

Stream je možné odchytávat již zmíněným volně dostupným programem VLC Player. Jeho konfigurace, znázorněná obrázkem číslo 5.5., je velice jednoduchá: Média->Otevřít síťový proud. Zde je nutné nastavit URL adresu včetně portu, na kterém je video streamováno. Adresa pro protokol UDP zní: *udp://@:port*, kde znak zavináče vyjadřuje právě adresu broadcastu.

5.1.2 Stream pomocí Youtube

Youtube nabízí zdarma možnost živého přenosu, ale vzhledem k tomu, že tento streamovací server přepočítává vstupní rozlišení do několika dalších, kontroluje autorská práva a provádí mnoho dalších operací, vzniká zpoždění odchytávaného signálu oproti vysílanému zhruba 40 sekund. Tento způsob streamu je tedy naprosto nevhodný pro chod této aplikace. Avšak lze na něm zdarma testovat nastavení FFmpeg modulu a kontrolovat tak kvalitu odchozího signálu.

Při nastavení streamu na serveru youtube, dostává uživatel název streamu a adresu primárního a záložního serveru, kam může signál posílat. Oproti VLC přijímá youtube data pouze pomocí protokolu rtmp, stejně jako většina streamovacích serverů.

Zápis FFmpeg příkazu pro stream může vypadat takto:

>>ADD 1 STREAM "rtmp://a.rtmp.youtube.com/live2/nazev_streamu" -y -f v4l2 -f jack -flags +global_header -acodec libvo_aacenc -ac 2 -ar 44100 -ab 128k -vcodec libx264 -g 2 -b 4500k -preset:v ultrafast -f flv

Příkaz obsahuje argumenty pro kódování zvuku, videa a mnoho další nastavení například datového toku. Každý streamovací server má jiné požadavky na příchozí stream, proto je dobré pro správné nastavení streamu si přečíst dokumentaci FFmpegu.[4]

5.2 Streamovací protokoly

5.2.1 UDP

Streamování je proces, při kterém je potřeba ignorovat chyby vznikající po cestě. Z toho důvodu byl roku 1980 Davidem P. Reedem navržen transportní protokol UDP (User Datagram Protocol), jako alternativu k protokolu TCP (Transmission Control Protocol). Oba jako součást protokolů TCP/IP. TCP zachovává relativně spolehlivý přenos na síťové vrstvě, avšak nevýhodou pro stream je posílání paketů po částech a ne po jedné cestě. Z toho vyplývá, že pakety mohou k cílovému uživateli přicházet zpožděné a tudíž neseřazené. Navíc poškozená data se musí přenášet znovu. Z toho důvodu je pro stream výhodnější protokol UDP, který sice zachovává nespolehlivý přenos dat, ale zato jsou data posílána seřazená a tudíž je pak protokol může rovnou ukládat do svých diagramů. Tím se stává přenos rychlejším.[9],[10]

5.2.2 RTP

Jako alternativa k protokolům TCP a UDP byl navržen protokol RTP (Real-Time Protocol), určen především pro streamování dat. Oproti TCP nabízí další datová pole určení pro časové značky a čísla sekvencí pro usnadnění načasování datového přenosu. Zároveň povoluje kontrolu serveru, díky které je video-stream posílán v taktu přizpůsobenému vykreslování obrazu v reálném čase. Přehrávač médií,

který podporuje přidaná pole RTP protokolu tak může shromažďovat přijaté pakety ve správném pořadí a rychlosti reprodukce obrazu.[10]

Během let vznikalo mnoho typů nástaveb tohoto protokolu, například protokol RTMP vytvořen firmou Macromedia, Inc., kterou poté roku 2005 koupil její dosavadní soupeř Adobe Systems, Inc.. Protokol, který po dlouhou dobu využívá pro stream dat například server Youtube, je určen především pro komunikace mezi serverem a Flashovým přehrávačem. Ten ještě ledna roku 2015 Youtube plně využíval, než ho nahradil nový HTML5 přehrávač. Protokol vycházející z principu TCP protokolu nabízí trvalé propojení a komunikaci s minimální latencí. Pro rychlý a hladký přenos jsou streamovaná data rozdělena do několika fragmentů, jejichž velikost je defaultně 64 bytů pro audio a 128 bytů pro video. Tato velikost však může být dle dohody obou stran dynamicky měněna a fragmenty prokládány. Přenos pak díky tomu lze multiplexovat.[11]

KAPITOLA

ŠESTÁ

ZÁVĚR

Cílem této bakalářské práce bylo navrhnout interaktivní multimediální aplikaci, která na základě požadavku koncového uživatele skrze sociální síť Twitter umožní změnu multimediálního obsahu. V rámci práce pak vytvořit klienta interpretujícího tento příspěvek v příkaz pro danou změnu obsahu pomocí přehrávacího programu CasparCG. V rámci této práce se podařilo docílit všech výše zmíněných bodů.

Klient nazvaný BPstream, který byl napsán pomocí programovacího jazyka Python, pracuje na třech hlavních vláknech, které má každé jasně danou úlohu. První vlákno řídí přehrávání samotného playlistu, přehrávání animací a zasílání všech příkazů z toho plynoucích do programu CasparCG. Playlist, který musí být ve formátu XML a obsahovat základní informace o videích, jako je jejich název, adresa a délka, je aplikací načten a uložen do jednotlivých polí SQLite databáze. Tato databáze je pak propojovací jednotkou všech vláken a vlákna tak spolu mohou skrze ní komunikovat. Druhým vláknem je řídící jednotka veškeré komunikace s Twitterem. Pomocí volně dostupné knihovny Twython tak toto vlákno přijímá streamovaná data Twitteru v daný čas přidaná formou příspěvků obsahujících předem definované klíčové slovo, zvaný hashtag. Díky možnosti streamu dat lze tyto příspěvky získávat se zpožděním v řádu pouhých desítek až stovek milisekund. Veškerá data jsou následně analyzována a filtrována podle toho, zda obsahují další hashtagy určující typ příspěvku. Příspěvkem je možné provést tři různé typy operací, a to buď hlasování o pořadí přehrávání videí, hlasování o kvalitě aktuálně přehrávaného videa a možnost přidání komentáře. Veškerá takto získaná a filtrovaná data se ukládají do již zmíněné databáze. Posledním vláknem je řízeno vypisování

komentářů do programu. Veškerá komunikace s programem CasparCG probíhá pomocí telnetu a protokolu AMCP. Z toho důvodu byla vytvořena knihovna těchto příkazů, skrze kterou všechna vlákna s programem komunikují.

Výstup programu CapsarCG lze nakonfigurovat pro video-stream na určitý streamovací server. V tomto ohledu se podařilo data streamovat na server Youtube, který tuto službu nabízí zdarma. Jediná jeho nevýhoda, a v případě této práce poměrně velká, je vysoké zpoždění výsledného obrazu. Proto je dobré tuto službu používat převážně pro testování a pro ostrý provoz aplikace používat server vlastní, či jiný placený, který nebude, oproti Youtube, streamované video převádět do více rozlišení. Při použití vhodného streamovacího serveru tak lze docílit zpoždění v rámci jednotek sekund až stovek milisekund.

Tato bakalářská práce sice splňuje veškeré body svého zadání, avšak lze ji v mnoha směrech dále rozvíjet. Tuto aplikaci je postupem času možné rozšířit i o možnost interpretace příspěvků přidaných pomocí dalších známých sociálních sítí jakými jsou například Facebook, Instagram, Google+. Stejně tak si lze vytvořit webové rozhraní pro přidávání příspěvků vlastní. Dále je možné více využít možností API sociální sítě Twitter a mimo pouhý text příspěvku a jméno autora vypisovat i jeho polohu či profilový obrázek. V případě sítě Instagram pak fotky vložené v příspěvku.

Během vytváření této bakalářské práce jsem si osvojil práci s novými technologiemi streamu a videem pomocí programu CasparCG, možnostmi API sociálních sítí a jejich dalším využitím v rámci vytvořené aplikace.

LITERATURA

[1] Chrástek Z.: *Kinoautomat - vznik a uvedení prvního českého interaktivního filmu.* Bakalářská práce Masarykova univerzita, Filosofická fakulta, 2010.

[2] Hůsek T.: *Vznik, vzestup, pád a znovuzrození interaktivního filmu.* Bakalářská práce Masarykova univerzita, Filosofická fakulta, 2013.

[3] Webové stránky programu CasparCG [online]. [cit. 2015-05-10]. Dostupný z WWW: `http://www.casparcg.com/`.

[4] Webové stránky programu FFmpeg. [online]. [cit. 2015-05-10] Dostupný z WWW: `http://www.ffmpeg.org/`

[5] Webové stránky Twitter API. [online]. [cit. 2015-05-10] Dostupný z WWW: `https://dev.twitter.com/`

[6] Kevin Makice: *Twitter API: Up and running: Learn how to build applications with the Twitter API.* „O'Reilly Media, Inc.“, 2009

[7] Wiki stránky programovacího jazyka Python. [online]. [cit. 2015-05-10] Dostupný z WWW: `https://wiki.python.org/`

[8] Stránky knihovny Twython. [online]. [cit. 2015-05-10] Dostupný z WWW: `https://twython.readthedocs.org`

[9] Peterka Jiří: *TCP a UDP* (1999). [online]. [cit. 2015-05-21] Dostupný z WWW: `http://www.earchiv.cz/anovinky/ai1864.php3`

[10] Austerberry David: *The technology of video and audio streaming.* Taylor & Francis, 2005.

[11] Článek: *Real Time Messaging Protocol* [online]. [cit. 2015-05-21] Dostupný z WWW: `http://en.wikipedia.org/wiki/Real_Time_Messaging_Protocol`