VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

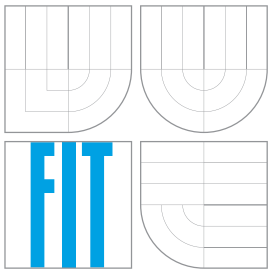
TEXTOVÝ EDITOR V PROSTŘEDÍ FLASH

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JIŘÍ PEJLA
AUTHOR

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

TEXTOVÝ EDITOR V PROSTŘEDÍ FLASH

FLASH-BASED TEXT EDITOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JIŘÍ PEJLA
AUTHOR

VEDOUCÍ PRÁCE — Ing. ZDENĚK VAŠÍČEK
SUPERVISOR

BRNO 2011

Abstrakt

Tato práce se zabývá rozborem textových editorů integrovaných do webového prohlížeče a tvorbou nového open-source editoru v prostředí Flash. Pro implementaci je využit multiplatformní jazyk haXe a nová knihovna dostupná ve frameworku Flash od verze 10 – Flash Text Engine. Výsledkem je snadno rozšiřitelná komponenta, která umožňuje označování, editaci a základní formátování textu a práci s obrázky. Editor pracuje s dokumenty ve formátu XHTML 1.0 Transitional.

Abstract

This bachelor's thesis deals with the implementation of a new open-source Flash-Based Text Editor. The theoretical part describes the most leading text editors integrated into the web browser. The source code of the component is written in multi-platform language haXe and the new Flash library – Flash Text Engine – is used. The result of the thesis is an easily extensible application that enables the user to select, edit and format texts and manipulate with pictures. The editor operates with documents in XHTML 1.0 Transitional format.

Klíčová slova

WYSIWYG editor, textový editor, haXe, Flash, ActionScript 3.0, Flash Text Engine, FTE, XHTML, XML Range

Keywords

WYSIWYG editor, text editor, haXe, Flash, ActionScript 3.0, Flash Text Engine, FTE, XHTML, XML Range

Citace

Jiří Pejla: Textový editor v prostředí Flash, bakalářská práce, Brno, FIT VUT v Brně, 2011

Textový editor v prostředí Flash

Prohlášení

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

.....
Jiří Pejla
17. května 2011

Poděkování

Rád bych poděkoval panu Ing. Zdeňku Vašíčkovi za konzultace, odbornou pomoc a poskytnutí knihoven pro rozhraní aplikace. Také chci poděkovat všem, kteří se podíleli na korekturách textové části této bakalářské práce.

© Jiří Pejla, 2011.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Kapitola 1

Úvod

Cílem této bakalářské práce je implementovat open-source editor do přehrávače Flash Player, který by umožňoval jednoduché formátování textu a práci s obrázky. Výstupem komponenty má být text ve formátu XHTML, případně XML. Do internetových prohlížečů již sice bylo implementováno mnoho editorů, ale většina je komerčního charakteru, jsou nekompatibilní napříč různými prohlížeči nebo neumožňují práci s obrázky. Proto je tato práce přínosem pro českou i zahraniční internetovou komunitu.

Práce je členěna následovně. První kapitola obsahuje detailnější specifikaci a stručné seznámení s problematikou. Dále následuje srovnání již existujících editorů v prostředí Flash i Javascript, popis jejich výhod a nevýhod. Teoretickým východiskem pro implementaci je kapitola 4, která popisuje použité technologie. Nejvíce pozornosti je vyhrazeno knihovně Flash Text Engine, která tvoří základ celého projektu. Další kapitola se zabývá návrhem aplikace – výběrem technologie a rozborem jednotlivých úkolů editoru. Popis implementace a testování je v kapitole 6. Její důležitou součástí je také popis aplikačního rozhraní. O výsledcích práce a dalším možném rozvoji pojednává kapitola 7. Závěr shrnuje cíle a výsledky celé práce.

Kapitola 2

Specifikace cílů

Jak již bylo zmíněno v úvodu, cílem práce bylo vytvořit komponentu, která bude umožňovat editaci textu zadaného ve formátu XML. Implementována měla být ve svobodném jazyce haXe, jehož překladač umožňuje vytvořit přímo ze zdrojového kódu soubor typu SWF, který lze spustit v přehrávači Flash. Vytvořit editor v tomto prostředí by nebylo možné bez použití nového textového engine (viz kapitola 4.1.2), jež je dostupný ve frameworku Flash od verze 10.

Výsledkem neměl být robustní program, který by obsahoval veškeré editační nástroje, ale jednoduchý a snadno rozšiřitelný základ textového editoru. Před započetím práce bylo nutné se seznámit se základy XML, s prostředím jazyka haXe, ale zásadní byla znalost skriptovacího jazyka ActionScript (v aktuální verzi 3.0) a jeho součásti – Flash Text Engine.

Flash Text Engine je nízkoúrovňová knihovna pro vytváření a ovládání řádků s textem. Neobsahuje hotové nástroje, které by umožňovaly označování a editaci, a proto bylo nutné vytvořit vlastní metody pro tyto funkce. Další částí projektu byla práce s dokumentem XML. ActionScript obsahuje knihovny pro práci s XML stromem pomocí DOM (Document Object Model [6]). Ty umožňují rozdělení (parsování) dokumentu do elementů textového engine. Není zde však podpora pro XML Range (viz [10]), který je potřeba pro obalení vybraného textu nějakým elementem (např. při změně typu písma na kurzívu), nebo pro odstranění výběru. Součástí projektu bylo tedy vytvoření vlastní třídy pro Range.

Důležitým požadavkem na komponentu je práce s obrázky – přesouvání, změna velikosti a zarovnání. S využitím DOM se dají snadno přesouvat uzly XML a měnit jejich atributy. Správné zobrazení je potom otázkou dobré implementace parsování dokumentu.

Kapitola 3

Textové editory v prohlížečích

Cílem projektu je implementovat editor v prostředí Flash, proto uvádím přehled nejvýznamnějších dostupných řešení a jejich výhody a nevýhody. Kromě editorů postavených na platformě Flash zmíním pro lepší představu také komponenty založené na Javascriptu. Informace o jednotlivých aplikacích jsem získal ze stránek projektů, které jsou uvedeny v každé podkapitole.

3.1 Editory založené na Javascriptu

3.1.1 CKEditor

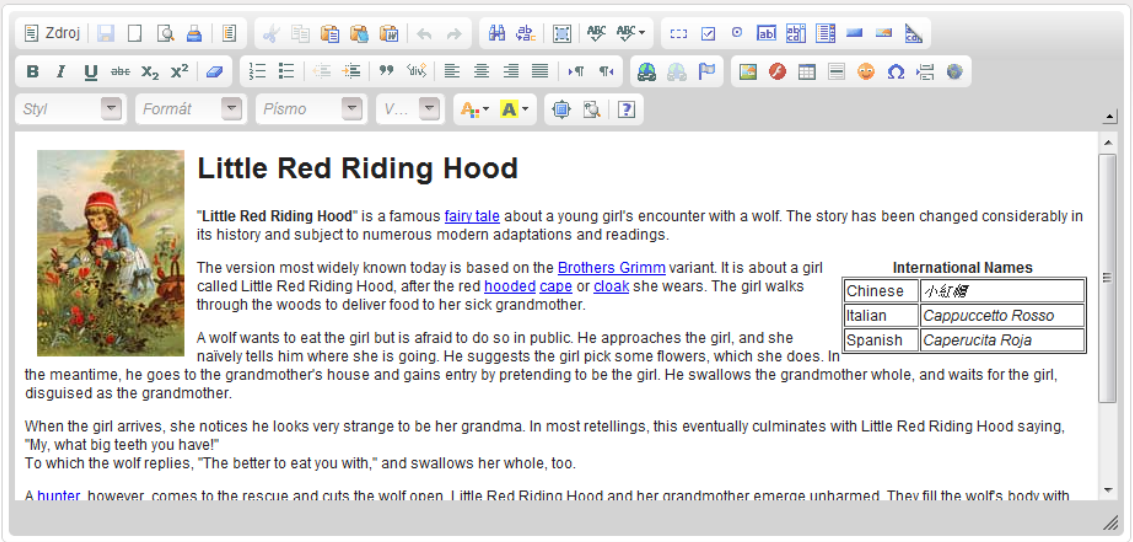
CKEditor je open-source komponenta, proto lze jeho rozhraní libovolně měnit a může být integrován do webových stránek. Kromě open-source licence je distribuován také pod komerční licencí CDL a podporuje všechny nejrozšířenější internetové prohlížeče. Pracuje s dokumenty ve formátu XHTML a umožňuje také integraci souborů Flash. Obsahuje všechny běžné editační nástroje pro práci s textem, obrázky a tabulkami. Profesionální technická podpora je zpoplatněna. Vyzkoušet si tuto komponentu je možné na stránkách http://ckeditor.com/demo. Podobným editorem je také TinyMCE, který je integrován do mnoha CMS (Content Management Systems).

3.1.2 Aloha Editor

Tento editor je založen na Javascriptu a použití HTML 5 a vlastnosti „contenteditable“. Nabízí editační nástroje pro text, tabulky i obrázky, podporuje DOM a je distribuován pod open-source i komerční licencí. Je možné jej integrovat do webových stránek a obsahuje API pro rozšíření (pluginy). Další informace jsou dostupné na adrese http://www.aloha-editor.org/.

3.1.3 WYMeditor

WYMeditor je jednoduchá open-source komponenta pro editaci dokumentů XHTML, která není zaměřena na vizuální vzhled, ale na strukturu dokumentu. Patří do skupiny WYSIWYM (What You See Is What You Mean) editorů, obsahuje podporu kaskádových stylů (CSS) a je integrován do mnoha CMS. Podporuje seznamy i tabulky, ale je omezena práce s obrázky. Stránky projektu jsou http://www.wymeditor.org/.



Obrázek 3.1: Vzhled propracované Javascriptové komponenty CKEditor

3.1.4 Výhody a nevýhody

Většina editorů založených na technologii Javascript podporuje velké množství funkcí i pokročilou práci s obrázky. Jejich nevýhodou však je, že každý podporuje jinou podmnožinu webových prohlížečů. V případě Aloha Editoru navíc ne všechny prohlížeče plně podporují novou technologii HTML 5.

3.2 Editory v prostředí Flash

3.2.1 Adobe Buzzword

Adobe Buzzword je robustní editor, který poskytuje veškeré běžné editační nástroje pro práci s obrázky, tabulkami atd. Jeho vstupem jsou dokumenty aplikací Microsoft Word, Open Office nebo formát RTF, výstupy mohou být ve formátu HTML a PDF. Rozhraní editoru umožňuje pokročilé sdílení a úpravy dokumentů. Je to komerční nástroj, pro používání vyžaduje registraci. Díky tomu je ale dostupná profesionální technická podpora. Účet s omezenými možnostmi je zdarma. Celý tento projekt je dostupný na stránkách https://acrobat.com.

3.2.2 Obedit

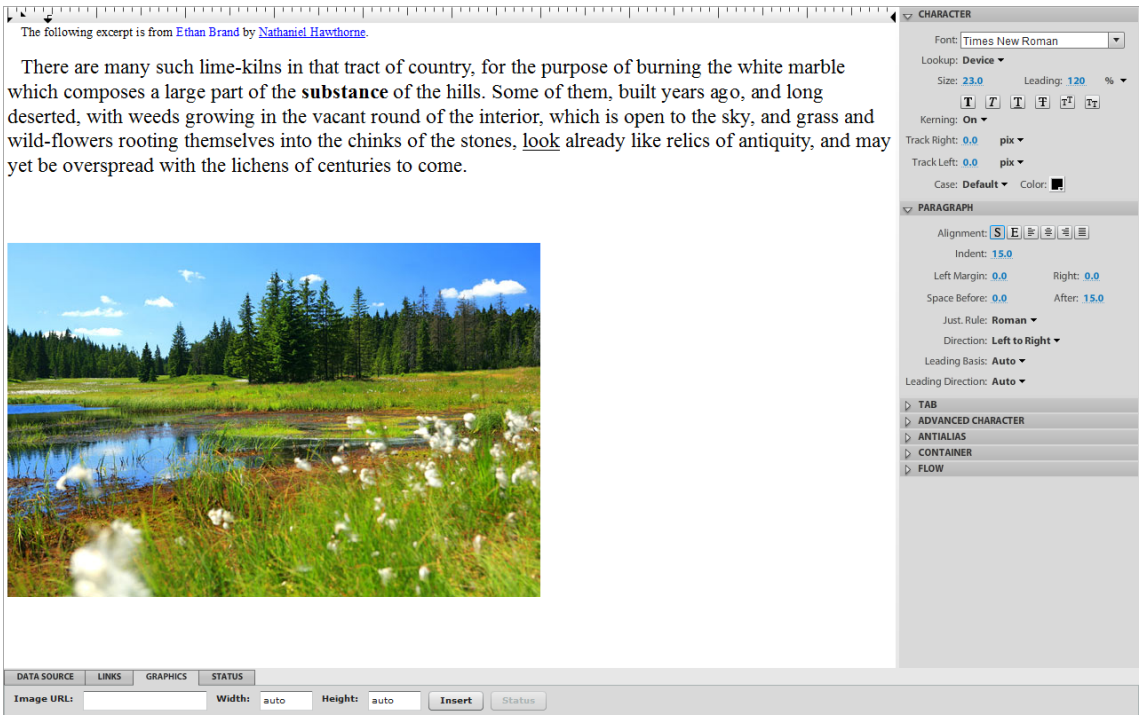
Tento editor je založen na prostředí Flash a je distribuován pod licencí LGPL. Zvládá běžné editace textu, ale není v něm možné pracovat s obrázky a tabulkami, protože používá pouze třídu Flash TextField. Pracuje jen se staršími verzemi HTML formátu, což už není v dnešní době vyhovující. Editor je k dispozici na webu http://www.oblius.com/projects/obedit/.

3.2.3 Flash Text Editor

Tato komerční komponenta je implementována nad textovým polem ve frameworku Flash. Zvládá základní editaci textu i použití kaskádových stylů. Obrázky je možné pouze vkládat, což je jeho velká nevýhoda. Obsahuje správce souborů. Více informací i demo komponenty jsou dostupné na http://www.flashtexteditor.com/.

3.2.4 Text Layout Editor

Text Layout Editor je ukázková aplikace nového Text Layout Frameworku (TLF). Obsahuje velké množství funkcí, ale nepodporuje plně formát XHTML a není možné zasahovat do jeho zdrojového kódu. Protože TLF je stále ve vývoji, má tento nástroj ještě mnoho nedostatků. Editor je k dispozici k vyzkoušení na http://labs.adobe.com/technologies/textlayout/demos/.



Obrázek 3.2: Ukázka rozhraní komponenty Text Layout Editor

3.2.5 WYSIWYG editor

Tato open-source komponenta je výsledkem bakalářské práce studenta Českého vysokého učení technického v Praze Bc. Lukáše Němečka pod vedením Ing. Martina Mudrocha (viz [8]). Aplikace zpracovává validní dokumenty XHTML a obsahuje konvertor mezi formáty HTML a XHTML. Je napsána v Adobe Flex frameworku s využitím komerčního IDE – Flash Builderu. Umožňuje základní editaci textu a práci s kaskádovými styly, ale nezvládá pokročilejší práci s obrázky. Editor je možné rozšířit pomocí pluginů.

3.2.6 Výhody a nevýhody

Použití technologie Flash je výhodné v tom, že zajišťuje úplnou přenositelnost z hlediska webových prohlížečů, protože zahrnuje jeden integrovaný přehrávač a velká většina uživatelů má nainstalovánu nejaktuálnější verzi.

Základní nevýhodou většiny těchto editorů je, že to jsou komerční nástroje, nebo nejsou open-source, a proto není možné si je více přizpůsobovat a rozšiřovat jejich zdrojový kód. Dalším problémem je práce s obrázky. Některé editory tuto možnost vůbec nemají, nebo podporují pouze změnu velikosti či zarovnání, ale přesun obrázku (např. pomocí myši) neimplementují.

Kapitola 4

Použité technologie

Následující kapitola je věnována technologiím v prostředí Flash, které jsem mohl ve své práci použít. Informace o jednotlivých nástrojích byly získány z [1, 5, 7, 11].

4.1 Flash

Flash je rozsáhlý framework od společnosti Adobe pro tvorbu interaktivních animací, webových aplikací a her. Má vlastní programovací jazyk, který vznikl (stejně jako jazyk Javascript) z ECMAScriptu – ActionScript v aktuální verzi 3.0. Také má vlastní vývojové prostředí (IDE) – Adobe Flash Professional Creative Suite (CS, v aktuální verzi 5.5). Cílové soubory jsou ve formátu SWF nebo EXE (spustitelné soubory pro Windows, které v sobě mají zabudovaný přehrávač Flash Player).

Co se týče textu, původně byla ve Flash pouze třída TextField (od verze 6), která byla vhodná pouze pro krátké, statické texty. Na tuto třídu navazovala třída TextArea. Od verze 10 přehrávače Flash Player je do něj zabudována nová knihovna pro rozsáhlejší texty – Flash Text Engine. Nad ní je vybudován rozsáhlý Text Layout Framework.

4.1.1 TextArea

TextArea je třída pro Flex, která umožňuje označování a editaci většího množství textu. Dokáže zpracovat text v jazyce HTML (základní typy elementů) a jeho formátování pomocí podmnožiny vlastností kaskádových stylů. Neobsahuje žádné rozhraní pro změnu formátování textu. Nyní jsou dostupné dvě verze této třídy.

Starší třída TextArea z balíčku mx.controls je založena na třídě TextField. Dokáže zobrazit obrázky, ale nepodporuje práci s nimi. Neumožňuje ani zpracovávání dokumentu pomocí DOM, proto nelze například navázat uživatelskou akci na kliknutí na obrázek apod.

Druhá, novější TextArea pro Flex 4 se nachází v balíčku spark.components. Je napsána pomocí Text Layout Framework a používá jeho objektově-orientovaný model formátovaného textu (viz kapitola 4.1.3). S objekty TLF může programátor libovolně manipulovat. TextArea obsahuje aplikační rozhraní pro navázání komponent na formátování textu. Nabízí i lepší typografické vlastnosti než starší verze a podporu pro více světových jazyků.

4.1.2 Flash Text Engine

Flash Text Engine je nízkoúrovňová knihovna pro tvorbu a zobrazení většího množství textu. Má v tomto nahradit původní třídu TextField. Podporuje většinu světových jazyků

a znaků.

Knihovna se stará pouze o tok textu – zobrazení a zalamování řádků s textem a obrázky. Neobsahuje nástroje pro změnu vzhledu (odsazení, odrážky, zarovnávání obrázků atd.) ani pro některé formátování textu (např. podtržené nebo přeškrtnuté písmo, barevné pozadí). Také neimplementuje označování. Všechny tyto věci nezpůsobují nové zobrazení textu, tzn. mohou být implementovány zvlášť, a proto je jejich naprogramování ponecháno na vývojářích. FTE nabízí pouze podporu v podobě zachytávání událostí nad textem a jejich zrcadlení do elementů této knihovny.

FTE je založen na modelu MVC (Model-View-Controller), proto je potřeba znát jen několik základních tříd:

- TextBlock – slouží pro uložení obsahu odstavce; jeho vstupem jsou objekty třídy ContentElement a výstupem řádky určené šířky s textem a obrázky; patří do řídící části (FTE Controller),
- ContentElement – abstraktní třída, která udržuje textový nebo grafický obsah třídy TextBlock; může přijímat události od TextLine pomocí třídy EventDispatcher; patří do části model (FTE Model); má tři podtřídy:
  - TextElement – uchovává text a jeho formátování; text může být změněn metodou `replaceText`,
  - GraphicElement – uchovává jakýkoliv zobrazitelný objekt (třída DisplayObject),
  - GroupElement – může obsahovat objekty TextElement, GraphicElement nebo jiné objekty GroupElement, je základem stromové struktury FTE, umožňuje operace nad svými prvky jako odstraňování, spojování atd.,
- ElementFormat – formátování obsahu některé z podtříd třídy ContentElement,
- FontDescription – vlastnosti fontu aplikované na třídu ElementFormat,
- TextLine – jediná zobrazitelná třída (FTE View), která se skládá z atomů (znaky, grafické a neviditelné elementy); detekuje události od uživatele a jedinou možností, jak ji vytvořit, je zavolat metodu příslušného objektu TextBlock; obsahuje metody pro získávání informací o jednotlivých atomech; všechny řádky v objektu TextBlock jsou svázány jako obousměrný seznam; řádek má vlastnost „validity“, jenž určuje, zda byl změněn objekt ContentElement, který je daným řádkem reprezentován

Události nad textem je možné sledovat vzhledem k řádkům nebo objektům ContentElement. První způsob je vhodný například pro vykreslování označení. Druhý způsob se realizuje pomocí třídy TextLineMirrorRegions, která zrcadlí události nad řádky do elementů FTE. Díky tomu je možné zjistit, které elementy řádek zobrazuje a nad kterým z nich byla vyvolána událost. Příkladem využití je implementace funkčnosti odkazů.

Pro lepší pochopení aplikace této knihovny uvádím velmi jednoduchý kód v jazyce ActionScript 3.0, který vygeneruje jeden odstavec textu s odsazeným prvním řádkem:

```
var str = "Hello World! This is quite simple
   Flash Text Engine example! Just look at it and
   think about it. It is not so difficult.";
var format:ElementFormat = new ElementFormat();
var text:TextElement = new TextElement(str, format);
```

```
var block:TextBlock = new TextBlock();
block.content = text;

var y:Number = 0;
var line:TextLine = block.createTextLine(null, 185);
line.x = 15;
while (line)
{
   addChild(line);
   y += line.ascent;
   line.y = y;
   y += line.descent;
   line = block.createTextLine(line, 200);
}
```

Z příkladu je patrné, jak se vytvářejí jednotlivé řádky z objektu třídy TextBlock. Odsazení řádků a mezery mezi nimi se určí nastavením jejich pozice (souřadnic). Stejným způsobem se vytváří zarovnání textu na střed, doprava nebo vícesloupcový vzhled – stačí správným výpočtem určit šířku a souřadnice řádku. Řádek zarovnaný do bloku musí být již takto vygenerován z objektu TextBlock, který má správně nastavenu vlastnost „textJustifier“. Je možné získat polohu a rozměry jednotlivých atomů v řádku, ale nelze tyto údaje měnit.

Implementovat editaci textu v FTE je možné pouze tak, že se změní obsah určitého elementu, nebo se vytvoří elementy nové, původní řádky se odstraní ze zobrazení a znovu se vygenerují z objektu TextBlock. Generovat všechny řádky znovu po každé editaci by bylo velmi náročné, proto při změně elementů v objektu TextBlock jsou zasažené řádky označeny jako nevalidní. Blok má také užitečnou vlastnost „firstInvalidLine“, která obsahuje odkaz na první nevalidní řádek v odstavci. Po editaci se tedy mohou znovu vytvořit pouze ty řádky, které byly změněny.

Vytváření řádků je náročné na paměť i procesorový čas, proto má třída TextBlock metodu, která použije data ze starého řádku při tvorbě nového. Je tedy výhodné si odstraňované řádky ukládat a znovu je používat.

Zobrazování obrázků je jednoduché, pokud jsou bez zarovnání. Jsou součástí řádku a mohou být umístěny mezi textem. Spodní okraj obrázku je zarovnán se spodním okrajem řádku a výška řádku je upravena podle výšky obrázku. Zarovnané obrázky musí být posunuty a u dalších obtékajících řádků je nutné upravit šířku a souřadnice.

4.1.3 Text Layout Framework

Tato open-source knihovna je vybudována nad Flash Text Engine a je celá napsána v jazyce ActionScript 3.0. Je také založena na modelu MVC a skládá se ze tří komponent: core, conversion a edit. Podporuje vertikální text, asijské jazyky a obsahuje aplikační rozhraní pro práci s textem.

Framework reprezentuje text pomocí hierarchického stromu, kde kořenem je objekt třídy TextFlow a ostatní prvky jsou podobné elementům jazyka XHTML. Jsou zavedena přísnější pravidla pro umístění elementů než jaká jsou v XHTML. Dále je možné jednoduše využít ukládání operací nad textem a implementovat tak akce „zpět“ a „znovu“. Podporuje také kopírování a vkládání. Importovat lze prostý text nebo XML v určeném tvaru.

Knihovna umožňuje jednoduché použití kaskádových stylů a vkládání obrázků a odkazů. Framework je však stále ve vývoji a obsahuje mnoho chyb (viz [8]). Problémy se vyskytují s importem nepodporovaných značek, s výstupem, který obsahuje zbytečné tagy a atributy, s obtížnou prací s obrázky (obtékání nelze jednoduše realizovat) a s odrážkami a číslováním.

4.2 HaXe

HaXe je multiplatformní, open-source programovací jazyk. Jeho cílové platformy jsou: Javascript, Flash, NekoVM, PHP a C++. Je to silně typovaný jazyk, obsahuje regulární výrazy i objektově orientované rysy. Podporuje debuggování a ladění. K funkčnosti překladače Flash přidává inlinování a šablonování. Umožňuje překlad do zdrojového kódu programovacího jazyka ActionScript 3.0 nebo přímo do formátu SWF (formát pro přehrávač Flash Player) a tento výstup je plně kompatibilní s technologiemi Flex a Flash. Více informací je dostupných na [3].

Kapitola 5

Návrh řešení editoru

5.1 Výběr technologie

Vzhledem k nejlepší přenositelnosti mezi různými webovými prohlížeči byla zvolena technologie Flash. V zadání bylo uvedeno, že je vyžadováno použití open-source prostředí. Jednou z možností bylo použití Adobe Flex, který má širokou podporu mezi vývojáři. Pro vytváření aplikací ve Flexu je ale vhodné využít komerční IDE – Adobe Flash Builder. Místo toho se nabízela možnost pokrokového open-source jazyka haXe, který umožňuje jednoduchý překlad kódu přímo do formátu pro přehrávač Flash Player. Pro jazyk haXe je na internetu dostupná kvalitní dokumentace, jeho překladač je schopen vygenerovat zdrojový kód skriptovacího jazyka pro Flash (ActionScript v aktuální verzi 3.0) a má s tímto jazykem téměř shodnou syntaxi. Proto byl vybrán jako nejlepší varianta.

Dále bylo potřeba rozhodnout, zda bude použit Text Layout Framework nebo pouze nízkoúrovňová knihovna Flash Text Engine. Jedním z hlavních cílů projektu byla práce s obrázky. Bc. Lukáš Němeček se ve své bakalářské práci ([8]) pokusil vytvořit editor nad Text Layout Frameworkem a nepodařilo se mu implementovat úpravy obrázků. Kromě toho je tento framework stále ve vývoji. Vybral jsem si použití Flash Text Engine, i když bylo nutné implementovat vše od základních operací s textem. Ale nebyl jsem omezován chybami a nedostatky stále se vyvíjejícího frameworku.

5.2 Úkoly editoru

Kapitola je věnována analýze jednotlivých částí editoru. Vychází ze základních znalostí knihovny Flash Text Engine (viz kapitola 4.1.2) a formátu XHTML.

5.2.1 Parsování dokumentu

Vstupem editoru je validní dokument XML (XHTML 1.0 Transitional), který musí být rozdělen podle elementů. Obsah každého odstavce musí být uložen do objektu třídy TextBlock. Celý text je tedy uložen v poli těchto bloků. Uvnitř odstavce mohou být další elementy (například „img“ pro obrázek). Ty jsou rozděleny do objektů třídy GroupElement uvnitř kterých mohou být další objekty GroupElement nebo přímo text (třída TextElement) či obrázek (třída GraphicElement). Elementy od sebe musejí dědit formátovací vlastnosti. Pro reprezentaci dokumentu pomocí knihovny FTE se tedy používá podobná stromová struktura jako v XML.

ActionScript 3.0 obsahuje třídy pro práci s XML pomocí DOM. Příslušná vlastnost třídy se nastaví na text dokumentu a potom se zavolá metoda, která provede parsování dokumentu XML. Následně je možné pracovat s XML stromem a jeho uzly a rozdělit tento strom do elementů knihovny FTE.

Další důležitou součástí je optimalizace XML stromu – odstranění prázdných elementů, spojení stejných sousedních elementů uvnitř odstavců atd.

5.2.2 Zobrazení textu

Text máme rozdělen do objektů třídy TextBlock a potřebujeme jej zobrazit (vyrenderovat). K tomu slouží metoda zmíněné třídy TextBlock – `createTextLine`, která vytvoří zobrazitelný řádek o zadané šířce. Obsahem řádku mohou být různě formátovaná písmena i obrázky. Zadáním souřadnic řádku určujeme odsazení a mezery mezi řádky. Po vytvoření všech řádků daného odstavce přejdeme na nový odstavec a y-ovou souřadnicí jeho prvního řádku určíme mezeru mezi odstavci.

Editor by měl umožňovat zobrazit podtržený a přeškrtnutý text. Toto formátování nelze nastavit danému elementu, ale musí se vykreslit zvlášť. Součástí zobrazování je tedy i vykreslení čar nad textem, který je takto formátován.

5.2.3 Označování textu

Text s obrázky je nyní zobrazen pomocí objektů třídy TextLine. Knihovna FTE neobsahuje nástroje, které by zajistily označování a editaci textu, proto je nutné vše od začátku implementovat.

Potřebné události můžeme navázat na objekt ContentElement, který je zrcadlí příslušnému řádku, nebo přímo na řádek. V projektu je výhodné znát, na který objekt ContentElement bylo například kliknuto, proto se naváží události na elementy FTE.

Při kliknutí na text se vyvolá událost a příslušná metoda se postará o nakreslení kurzoru a uložení polohy v textu. Polohou se myslí jednak pozice v rámci elementů FTE a jednak pozice v rámci XML stromu. Stačilo by ukládat pozici pouze v XML stromu, ale to by při každé drobné editaci bylo nutné celý text smazat, parsovat XML a zobrazit znovu, což by zbytečně zatěžovalo procesor. Drobné editace je možné optimalizovat tak, aby editace proběhla i v příslušném TextElementu (metoda `replaceText`) a znovu vytvořeny byly jen zasažené řádky.

Pro ukládání pozice v XML stromu jsem se inspiroval rozhraním XML Range definovaným webovým konsorciem (viz [10]). V projektu by bylo postačující definovat rozsah v XML stromu pouze pomocí počátečního textového uzlu a offsetu v něm a koncového textového uzlu a offsetu, protože se vždy označuje přímo text.

Výběr více znaků je nutné zvýraznit kreslením šedých obdélníků nad příslušné řádky. Při změně výběru se tedy nezobrazuje celý text znovu, pouze se změní nakreslené zvýraznění a uloží se aktuální poloha v textu a XML stromu. Samozřejmě je nutné speciálně ošetřit označování obrázků.

Označování zahrnuje i posuny kurzoru pomocí kláves – Home, End a šipek. Při držení klávesy Shift a stisknutí těchto kláves se musí pokračovat v označování.

5.2.4 Editace textu

Původním záměrem bylo provádět editace pouze v rámci elementů FTE a změny zrcadlit do XML stromu. To ale bylo zamítnuto z důvodu, že změn v textu může být velmi mnoho

a může se například i měnit odsazování odstavců, což by se velice těžce zjišťovalo v rámci řádků s textem. Také editace v rámci více elementů FTE je velmi těžké realizovat. Proto se všechny změny okamžitě zrcadlí do XML stromu.

Jak již bylo zmíněno v předchozí kapitole, jednoduché editace by se měly optimalizovat tak, že se provedou zároveň v elementu FTE i uzlu XML a znovu se vytvoří jen zasažené řádky. Mezi jednoduché editace patří psaní a odstranění znaku, pokud se označení týká pouze jediného XML uzlu a příslušného elementu FTE. Když je označeno více uzlů, změny se provedou pouze ve stromu XML, který je pak parsován a celý dokument je znovu zobrazen.

Aby editor poskytoval základní funkčnost, měly by fungovat následující klávesy:

- Klávesy pro psaní textu – pokud je označeno více znaků, jsou přepsány znakem novým, jinak vepíše znak do textu,
- Delete – pokud je označeno více znaků, odstraní výběr, jinak odstraní následující znak,
- Backspace – stejná funkčnost jako Delete, ale při označení kurzorem se nejprve provede posun kurzoru o jedno místo doleva,
- Enter – při označení více znaků, odstraní výběr; vytvoří nový odstavec v XML stromu a vyvolá renderování textu; při držení klávesy Shift pouze vypíše znak nového řádku

U klávesy Enter vznikají problémy, je-li stisknuta, když je kurzor uvnitř elementů vnořených do odstavce. V tomto případě je nutné uzly kopírovat, rozdělit a přesunout tak, aby uzly za kurzorem byly umístěny v nově vytvořeném odstavci.

Po provedení editace je důležité znovu označit příslušné místo v textu, aby se například mohlo dále pokračovat v psaní.

5.2.5 Práce s obrázky

Editor by měl umožňovat přesun, změnu velikosti a zarovnání obrázků. Označení obrázku by mělo být speciálně ošetřeno, aby se například při jeho odstranění znovu vyrenderoval celý dokument. Zjišťovat, na které řádky ještě obrázek zasahoval a na které ne, by bylo příliš náročné, proto je po každé úpravě obrázku zobrazen celý text znovu.

Přesouvání zarovnaného obrázku je možné udělat dvěma způsoby. Buď je možné obrázek přesunout pouze na začátek řádku, což sice odpovídá zobrazení v prohlížeči, ale neodpovídá to umístění v textu, protože text je pak vyrenderován jinak. Nebo lze obrázek umístit libovolně v textu, pak je přesunut přímo do patřičného místa, ale při zobrazování se musí zalomit odstavec, aby bylo patrné, kde přesně se obrázek nachází. Implementace prvního způsobu by byla velmi náročná, protože by se musely nejprve dopředu vygenerovat řádky a zjistit, kde je místo začátku řádku, a pak teprve obrázek umístit a vše vyrenderovat znovu. Proto jsem si vybral druhou možnost řešení. Nezarovnané obrázky mohou být umístěny kamkoliv do textu.

Uživatelsky nejpřívětivější je měnit velikost obrázku pomocí tažení myši. Kvůli tomu se na obrázek nakreslí do rohu trojúhelník, který bude reagovat na události myši. Důležité je, aby uživatel měl možnost zachovat původní poměr stran – k tomu se používá klávesa Shift. Změna velikosti se projeví tak, že se změní atributy příslušného uzlu „img“ a dokument je znovu zobrazen.

Základní hodnoty zarovnání obrázku jsou doleva, doprava a bez zarovnání. Kliknutím na příslušné tlačítko se změní atribut uzlu a dokument se znovu renderuje.

5.2.6 Změna formátování textu

Formátování se děje pomocí obalení vybraného textu určitým uzlem (např. „b“ nebo „i“). Pokud je označen text přes více odstavců nebo elementů, musí se vytvořit tolik formátovacích elementů, aby se vzájemně nepřekrývaly. Překrývání značek je v XML zakázáno.

Při označení více uzlů je možné projít všechny uzly mezi nimi a obalit je formátovacím uzlem. Když mají počáteční a koncový uzel stejného rodiče, mohou se uzly mezi nimi vložit do jediného formátovacího uzlu.

Důležitou vlastností je, že pokud je označený text již stejně naformátován, kliknutí na tlačítko jej formátování zase zbaví. Sporná situace nastane, je-li označený text naformátován jen částečně. Zde je nutné se rozhodnout, jestli jej naformátovat celý, nebo odstranit již vytvořené formátování.

Další možností, jak určit formátování, je změnit typ odstavce. Název odstavcového elementu, ve kterém se vyskytuje kurzor nebo část označení, je změněn podle vybrané hodnoty a dokument je znovu zobrazen. Se všemi nadpisy se zachází stejně jako s odstavci, jen výchozí formátování je jiné.

Kapitola 6

Implementace a testování

6.1 Jednotlivé třídy editoru

Kapitola obsahuje stručný popis jednotlivých tříd a jejich nejdůležitějších vlastností a metod. Při implementaci jsem se snažil co nejvíce využívat třídy jazyka ActionScript 3.0, aby byl zdrojový kód co nejblíže tomuto jazyku a ti, kdo jej znají, se nemuseli učit konstrukce jazyka haXe.

6.1.1 MyXMLDocument

Tato třída reprezentuje dokument XML, je potomkem třídy XMLDocument. Nejprve jsem zkoušel využít aktuální třídu v jazyce ActionScript 3.0 pro práci s XML, ale přísný překladač haXe nedokázal přeložit konstrukce, které byly nezbytné pro přiřazování hodnoty uzlu či odstranění uzlu. Proto jsou v projektu využity původní třídy XMLDocument a XMLNode.

V konstruktoru třídy se pouze inicializují potřebné vlastnosti. Parsování textového řetězce do stromu XML se provádí v metodě `setDocumentContent`. V celém projektu se tedy používá vždy jen jeden dokument. Před zobrazením textu je vždy volána metoda `reload`, která optimalizuje strom a potom jej rozdělí do elementů knihovny FTE. Optimalizace se týká spojování stejných sousedních uzlů (pokud to nejsou odstavce) a odstraňování prázdných uzlů. Kromě toho jsou vytvořeny speciální pomocné odstavce, pokud se narazí na obrázek zarovnaný doleva či doprava, aby bylo viditelné, kde přesně je obrázek umístěn. Zarovnaný obrázek musí být umístěn na začátku odstavce. Optimalizační metoda je implementována rekurzivně, protože iterační průchod stromem by byl v tomto případě velice obtížný. Při parsování dokumentu se kontroluje jeho základní struktura a pokud se narazí na neznámý element, je zobrazeno hlášení, které informuje o vzniklé chybě.

Rozdělování stromu do elementů knihovny FTE začíná vytvořením objektu TextBlock pro každý odstavec textu. Za odstavce jsou považovány i všechny druhy nadpisů, jen mají jiné výchozí formátování. Celý dokument je rozdělen do pole objektů TextBlock. Uvnitř těchto objektů jsou uloženy textové a grafické elementy ve stejné stromové struktuře jako v XML. Je to umožněno existencí třídy GroupElement, jejímž obsahem mohou být jak textové a grafické elementy tak i objekty stejné třídy. Domníval jsem se, že když přiřadím formát objektu GroupElement, že jej zdědí podřízené elementy, ale bohužel to takto nefungovalo. Proto jsem byl nucen implementovat parsování uzlu odstavce rekurzivně a předávat do každého volání objekt ElementFormat.

Když se při parsování narazí na uzel „img“, musí být obrázek načten do objektu DisplayObject pomocí třídy Loader. Dokud nejsou načteny všechny obrázky, nemůže proběhnout

zobrazení dokumentu. Proto tato třída vytvoří událost pro třídu LineContainer po skončení načítání. Na procesorový čas bylo velmi náročné načítání všech obrázků při každém parsování dokumentu. A tak jsem vytvořil pomocné pole obrázků a pokud byl obrázek již načten, použije se znovu. Po načtení všech obrázků se ještě musí nastavit jejich velikost podle atributů v příslušném XML uzlu.

Do každého elementu FTE se při parsování ukládá ukazatel na příslušný uzel XML, aby bylo možné jednoduše zjistit polohu v XML stromu při označování textu. Také se ukládají do vlastností třídy LineContainer textové elementy, které mají být formátovány jako podtržené nebo přeškrtnuté, aby příslušné čáry byly vykresleny po vytvoření řádků s textem.

Před uložením dokumentu se musí odstranit pomocné odstavce a na začátek dokumentu je vložena jeho deklarace (XML 1.0 s kódováním UTF-8) a definice jeho typu (XHTML 1.0 Transitional). Verze XHTML Transitional byla zvolena z důvodu, že verze Strict nepovoluje atribut „align“, který je použit pro určení zarovnání obrázků.

6.1.2 LineContainer

Třída sloužící pro zobrazení celého dokumentu, je potomkem grafického kontejneru Sprite. Před zahájením práce s touto třídou je kromě konstruktoru, kde se nastaví potřebné odkazy, ještě potřeba zavolat metodu `setXMLDocument`, která nastaví odkaz na aktuální dokument. Změna šířky zobrazených řádků je možná metodou `setTextWidth`.

Hlavní funkčnost obstarávají dvě metody – `createLines` a `reCreateLines`. První metoda odstraní všechny aktuální elementy FTE a nechá je znovu načíst z dokumentu XML metodou `reload` třídy MyXMLDocument. Poté jsou vytvořeny a zobrazeny řádky všech odstavců reprezentovaných objekty třídy TextBlock.

Při vytváření řádků jsou ukládány souřadnice a rozměry obrázků, na které se narazí. Podle nich je nutné upravovat šířku řádku, aby nepřekrýval obrázek, a odsazení podle atributu zarovnání v uzlu XML. Obrázky zarovnané doleva nebo doprava jsou vždy na začátku odstavce (a řádku). Je nutné je posunout tak, aby horní okraj byl na stejné souřadnici jako horní okraj řádku, do kterého obrázek patří. U zarovnání doprava se ještě musí přehodit poloha obrázku a příslušného řádku. Nezarovnané obrázky jsou součástí řádku a neposunují se – jsou zobrazeny stejně jako ve webovém prohlížeči. Do rohu každého obrázku se ještě musí nakreslit trojúhelník, který slouží pro změnu velikosti.

Metoda `reCreateLines` je založena na tom, že při změně obsahu elementů uvnitř objektu TextBlock jsou automaticky označeny řádky, které byly změnami ovlivněny, jako nevalidní. Tato metoda je volána pouze po malých editacích v textu (napsání či odstranění znaku) a nevyvolává nové parsování XML dokumentu. Řádky zasaženého odstavce jsou znovu vytvořeny až do jeho konce. Další odstavce většinou není potřeba znovu přegenerovávat, stačí je posunout. Výjimkou je, pokud další odstavec je vedle zarovnaného obrázku. V tom případě musí být také přegenerován, aby nebyly řádky posunuty tak, že by překrývaly obrázek.

Nastavením y-ové souřadnice řádku se určuje mezera mezi řádky i odstavci. Když se narazí na speciální odstavec, který se vyskytuje u zarovnaných obrázků, odstavcová mezera nesmí být vytvořena. Mezery se musí určovat vzhledem k velikosti písma aktuálního odstavce, protože u větších nadpisů by se řádky překrývaly.

Po zobrazení celého dokumentu je ještě nutné projít pole textových elementů, které jsou formátovány jako podtržený či přeškrtnutý text. Pole jsou naplňována při parsování dokumentu ve třídě MyXMLDocument. U každého elementu se projdou všechny řádky,

které jej reprezentují, a pod nimi (nebo přes ně) se vykreslí patřičná čára.

Při odstraňování elementů FTE a tedy i všech řádků se řádky ukládají do pole a při novém zobrazování jsou znovu použity kvůli úspoře paměti a procesorového času. K tomu slouží metoda třídy TextBlock – `recreateLines`.

6.1.3 Range

Třída Range obstarává veškeré označování textu, editaci a formátování. Dědí od třídy Sprite, protože přímo vykresluje kurzor a označení.

Základním prvkem pro určení pozice v textu jsou dva indexy – číslo odstavce a pozice v něm. Z indexů se dá získat odpovídající objekt TextElement a offset v něm nebo objekt GraphicElement. Jednotlivé elementy obsahují odkaz na XML uzel, který reprezentují. Offset v textovém elementu je stejný jako offset v XML uzlu a to jsou všechna data, která jsou potřeba pro vymezení označeného rozsahu.

Označuje se pomocí myši nebo klávesy Shift a šipek. Je také možné označovat zavoláním příslušných metod, které uloží všechny potřebné údaje a označení vykreslí. Funkcím se musí předat výše zmíněné indexy – odstavce a znaku. Je možné připojit objekt TextField (metoda `setPathText`) k funkčnosti zobrazení cesty od aktuálního XML uzlu ke kořeni. Přitom je zobrazována cesta náležící znaku za kurzorem, protože kurzor je vždy vykreslen před označený znak (atom v elementu).

Při vykreslování označení nedochází k novému zobrazení celého textu, pouze jsou vykreslovány obdélníky nad jednotlivými řádky. Výška obdélníků musí odpovídat výšce aktuálního textu. Řádky jsou v objektu TextBlock uloženy jako obousměrný seznam, proto se dá jednoduše dostat k předchozímu a následujícímu řádku. Když následující řádek neexistuje, vezme se další objekt TextBlock z globálního pole pro odstavce. Označování je implementováno tak, aby byl označen znak, jakmile se nad něj najede myší. Druhou možností bylo označit jej, až se kurzor vyskytne za ním, ale dle mého názoru je první způsob pohodlnější a častěji používaný.

Metody pro editaci a posun v textu pracují s již uloženými indexy. Při editacích většího označení je nutné odstranit vybrané uzly. To jsem implementoval tak, že jsem vytvořil pomocnou metodu, která v XML stromu najde následující nejbližší textový element. Ten je vymazán a tak se pokračuje, dokud se nenarazí na cílový textový uzel. Otcovské elementy, které neobsahují žádný uzel, jsou vymazány při optimalizaci XML stromu v metodě třídy MyXMLDocument. Pokud výběr přesahuje přes více odstavců, za poslední neoznačený uzel prvního odstavce jsou připojeny zbývající neoznačené uzly posledního odstavce. Tím dojde k obvyklému spojení odstavců.

Po dokončení větších editací se zavolá metoda třídy LineContainer – `createLines`. Po drobných editacích je to metoda `reCreateLines`. Obě metody po dokončení vytvoří událost, která způsobí vykreslení označení podle nastavených globálních proměnných (vlastností). Načítání textu s obrázky může trvat delší dobu, proto se čeká na tuto událost, aby mohlo být správně do textu vykresleno označení.

Problém s klávesou Enter (vytvoření nového odstavce) týkající se označeného textu vnořeného do více elementů byl řešen rekurzivně. Jenom díky zanoření až do listů stromu je možné určit, které elementy klonovat a které přemístit do původního nebo nového odstavce. Stejná metoda je použita i při přemisťování obrázku do zanořených elementů, protože u zarovnaných obrázků se vždy vytváří pomocný odstavec, a proto musí být obalující elementy kolem něj rozděleny. Také při kopírování, vyjímání a vkládání textu je použito rozdělování XML stromu. Je totiž nutné rozdělit elementy v označeném místě a uložit si je

kvůli zachování formátování.

Obalení výběru textu elementem zadaného názvu (metoda `coverRange`) funguje tak, že se zavolá metoda pro obalení části odstavce na všechny odstavce, kterých se označení týká. U odstavců se obalují jednotlivé uzly, nebo se vytvoří jeden formátovací uzel, do kterého se vloží všechny označené – to je možné, pokud počáteční a koncový uzel mají stejného otce. Při formátování jednoho textového XML uzlu můžou vzniknout až čtyři nové v závislosti na počátečním a koncovém offsetu. Pokud je počáteční offset větší než nula, musí se vytvořit nový uzel pro text, který předchází textu označenému. Stejně tak, pokud je koncový offset menší než délka obsahu uzlu. Pro označený text se také vytvoří textový uzel, který se ale musí vložit do nového formátovacího uzlu. Pro odformátovávání výběru jsou použity stejné metody.

Označení obrázku musí být speciálně ošetřeno. Pokud je vybrán obrázek, nemůže být označeno nic jiného. Vymazání obrázku pouze odstraní daný uzel XML a zobrazí dokument znovu. Při změně velikosti obrázku nebo jeho zarovnávání doleva či doprava se pouze změní atributy XML uzlu a nastaví se globální vlastnosti (index odstavce a index znaku), aby po novém renderování textu byl označen stejný obrázek. Vytváření pomocných odstavců kvůli zarovnaným obrázkům má na starost třída MyXMLDocument. Během odstraňování zarovnání je pomocný odstavec vymazán. Přesunout obrázek tažením myši je možné do formátovaného i neformátovaného textu i do nadpisů.

Změna velikosti obrázku se provádí tažením myši za levý dolní roh. Přitom je zobrazován obdélník, aby bylo názorně vidět novou velikost. Při přesouvání obrázku je označen zešednutím a na místa pohybu myši je vykreslován obyčejný kurzor, který určuje přesnou polohu obrázku.

6.1.4 Editor

Editor je třída, která vytváří objekty výše zmíněných tříd (MyXMLDocument, LineContainer a Range), reaguje na události od uživatele, načítá vstupní soubory a předává je dále ke zpracování, zachytává výjimky a obstarává výpis chybových hlášení. Dědí ze zobrazovacího kontejneru Sprite, protože spojuje zobrazení textu a jeho označení.

V konstruktoru je možné nastavit jméno inicializačního vstupního XML souboru, šířka a výška zobrazovaného okna s textem. Při překročení zadané výšky textu je přebývající text skryt a dá se k němu dostat pomocí rolovacího kolečka myši. Tato třída nezobrazuje žádná tlačítka ani posuvník, ale obsahuje metody, které slouží pro připojení ke tlačítkům i posuvníku, např. metoda pro otevření souboru, vymazání celého textu atd. Metody pro formátování většinou pouze zavolají funkci ze třídy Range.

Důležité jsou funkce, které reagují na události myši. Při kliknutí do textu jsou spočítány potřebné indexy odstavce a znaku a ty jsou předány metodě třídy Range. Stejně se to děje i při vykreslování většího označení. Pohybem myši lze také měnit velikost a umístění obrázku. Ve třídě jsou ukládány potřebné údaje o obrázku a stará se o vykreslení zvětšovacího obdélníku a označení v případě přesouvání. Změna atributů obrázku se nakonec provede pomocí metody třídy Range. Třída Editor má také na starosti rolování kolečkem myši. Výšku obdélníku, do kterého je dokument ořezáván, je možné nastavit v konstruktoru.

Dalším úkolem třídy je zpracovávat všechny stisknuté klávesy. Je to uděláno jednoduše tak, že podle kódu klávesy se zavolá příslušná metoda třídy Range, která se postará o změnu dokumentu i následné nové zobrazení.

6.1.5 Example

Třída Example je ukázkový příklad využití textového editoru. Programátorům je dána naprostá svoboda při vytváření vzhledu. Vytvořením instance editoru a jeho zobrazením je vykreslen pouze editovatelný text, ostatní nástroje jsou volitelné. Proto je možné tuto komponentu využít i jen jako pole pro text.

Vytvořením vlastních tlačítek a textových polí a jejich připojením k metodám editoru vznikají nástroje pro práci s textem. V příkladu jsem použil třídu pro tlačítka, kterou mi poskytl vedoucí bakalářské práce. Posuvník je převzat ze svobodného projektu „haXe minimalcomps“ (viz [9]). Popis aplikačního rozhraní je v následující kapitole.

Obrázek 6.1: Ukázka výsledné funkcionality editoru

6.2 Aplikační rozhraní

Tato kapitola je věnována stručnému popisu nejdůležitějších veřejných vlastností a metod jednotlivých tříd. Nejsou zde vyjmenovány metody třídy MyXMLDocument, protože hlavní třída Editor poskytuje dostatek funkcí pro práci s dokumentem.

6.2.1 Editor

Třída Editor poskytuje tři veřejné vlastnosti. První dvě jsou proměnné „container“ a „range“, které obsahují instance tříd LineContainer a Range a umožňují tak programá-

torovi používat veřejné metody těchto tříd. Poslední vlastností je aktuální výška textu celého dokumentu („textHeight“). Tu je možné využít pro implementaci posuvníku.

Následuje výčet metod aplikačního rozhraní.

Konstruktor

`new(textWidth:Int = 670, windowHeight = 500, f:String = "")` Konstruktor vytvoří pole pro editovatelný text. Může rovnou zobrazit vstupní soubor („f“) a nastavit šířku řádků („textWidth“) a výšku okna, do kterého bude dokument ořezáván („windowHeight“). Pokud není zadán vstupní soubor, je zobrazen výchozí inicializační dokument.

Nastavení parametrů editoru

`setFieldWidth(width:Int)` Tato metoda nastaví novou šířku textu (a tedy i okna editoru).

`setFieldHeight(height:Int)` Metoda, která upraví výšku okna pro text.

`setPathText(text:Dynamic)` Metoda, která nastaví předané textové pole pro výpis aktuální cesty v XML stromu. Předané pole může být libovolného typu, ale musí mít vlastnost „text“.

`setSelectBox(box:Dynamic)` Této metodě se předává odkaz na libovolnou komponentu pro výběr typu odstavce. Komponenta musí implementovat metodu `setSelectedValue`, která nastaví název aktuálního odstavcového elementu.

`setFormattingButtonType(button:Dynamic)` Metoda slouží pro nastavení typu formátovacích tlačítek. Je velmi důležitá, protože když nebude typ správně nastaven, výběr textu se ztratí při kliknutí na tlačítko a nebude možné nic naformátovat. Parametrem se předává odkaz na jedno z tlačítek.

`setScrollY(Y:Float)` Tato metoda posune text v okně podle předané y-ové souřadnice.

Práce s dokumentem

`getSelectedHTML():String` Tato metoda vrátí řetězec s vybraným textem ve formátu XHTML. Začátek textu není označen odstavcem, další odstavce však ano.

`getPlainSelectedText():String` Výstupem této metody je aktuálně označený text bez jakýchkoliv značek.

`setDocumentContent(content:String)` Metoda slouží k nastavení celého obsahu dokumentu pomocí předaného řetězce. Jestliže dokument nemá správnou strukturu, je zobrazeno chybové hlášení. Po úspěšné kontrole a parsování je dokument zobrazen.

`getDocumentContent():String` Výstupem této funkce je obsah dokumentu se všemi značkami a jeho deklarací a definicí. Výsledný řetězec je odsazen tak, jak je obvyklé u formátu XML.

`getUnformattedDocument():String` Narozdíl od předchozího případu je zde výstupem text celého dokumentu bez jakéhokoliv odsazení.

`loadDocument(name:String)` Tato metoda umožňuje načíst soubor XHTML se zadaným jménem a zobrazit ho.

Tlačítka pro práci s celým dokumentem

`openClicked(btn:Dynamic)` Metoda, která zobrazí dialog pro výběr vstupního souboru.

`saveClicked(btn:Dynamic)` Metoda pro zobrazení dialogu pro uložení výstupu aplikace.

`clearClicked(btn:Dynamic)` Funkce, která odstraní veškerý text i obrázky.

Tlačítka pro formátování textu

`boldClicked(btn:Dynamic)` Metoda pro obalení vybraného textu elementem „b“, který značí tučné písmo.

`italicsClicked(btn:Dynamic)` Metoda pro obalení vybraného textu elementem „i“, který značí kurzívu.

`underlineClicked(btn:Dynamic)` Metoda pro obalení vybraného textu elementem „u“, který značí podtržení.

`strikeClicked(btn:Dynamic)` Metoda pro obalení vybraného textu elementem „s“, který značí přeškrtnutí písma.

`strongClicked(btn:Dynamic)` Metoda pro obalení vybraného textu elementem „strong“, který značí zvýrazněný text. Tento element slouží pro logické formátování a takto označený text je zobrazen tučně.

`emClicked(btn:Dynamic)` Tato metoda také slouží pro logické formátování a zvýraznění textu. Obalení elementem „em“ způsobí přepnutí typu písma na kurzívu.

Tlačítka pro práci s obrázkem

`pictureLeftClicked(btn:Dynamic)` Tato metoda změní atribut „align“ vybraného obrázku na hodnotu „left“, tzn. obrázek bude zarovnán doleva.

`pictureRightClicked(btn:Dynamic)` Tato metoda změní atribut „align“ vybraného obrázku na hodnotu „right“, tzn. obrázek bude zarovnán doprava.

`pictureNoneClicked(btn:Dynamic)` Tato metoda odstraní atribut „align“ a v důsledku toho nebude obrázek zarovnán – bude umístěn na řádku mezi textem.

Změna typu odstavce

`changeParagraphType(type:String)` Nastaví název všech vybraných odstavcových elementů na hodnotu předanou parametrem.

6.2.2 Range

Přes vlastnost range třídy Editor je možné dostat se k veřejným vlastnostem a metodám třídy Range.

Třída Range obsahuje mnoho vlastností, které vymezují počáteční a koncový bod aktuálního označení. Rozsah v XML stromu je určen počátečním uzlem („startNode“) a offsetem v něm („startOffset“) a koncovým uzlem („endNode“) a offsetem („endOffset“). Pozice uvnitř elementů FTE se určuje pomocí pole čtyř celých čísel – index počátečního odstavce a znaku v něm a index koncového odstavce a znaku. Toto pole je nazváno „selected“ a vždy obsahuje žádné nebo právě čtyři čísla (kurzor na jedné pozici se značí dvěma stejnými dvojicemi čísel). Měnit obsah těchto proměnných je vhodné pouze zavoláním metod `startCursor` a `drawSelection`, protože tyto metody nastavují další důležité privátní vlastnosti.

Vlastnosti pojmenované „selected“ a číslo proměnné odpovídají prvkům v poli „selected“. Slouží k tomu, aby po vyvolání nového zobrazení dokumentu (metody třídy LineContainer) bylo zobrazeno patřičné nové označení. K nim patří také proměnná „draw“ typu

Bool, podle které se vykreslí buď kurzor nebo označení více znaků. Vlastnost „copyNode“ obsahuje všechny uzly aktuálně zkopírované do schránky obalené elementem „copy“.

Zmíním se o důležitých veřejných metodách třídy Range.

Označování

`startCursor(atomT:Int, TBindex:Int)` Tato metoda slouží k nakreslení kurzoru na pozici danou indexem znaku a odstavce. Uloží také do vlastností všechny důležité údaje o pozici.

`drawSelection(atomT:Int, TBindex:Int, shift:Bool = false, left:Bool = false)` Metoda, která nakreslí označení na další pozici a uloží potřebné údaje.

Práce s obrázky

`pictureMove(node:XMLNode)` Tato metoda přesune obrázek na aktuální pozici, lze ji však použít pro přesun libovolného uzlu.

`changePictureSize(rect:Rectangle)` Metoda, která nastaví atributy „width“ a „height“ aktuálně vybraného uzlu podle rozměrů předaného obdélníku.

Kopírování a vkládání

`copy(cut:Bool = false)` Tato metoda zduplikuje označené uzly a duplikovaný seznam uchová v atributu „copyNode“. Změnou parametru se dosáhne vyjmutí uzlů.

`paste()` Tato metoda odstraní všechny označené uzly a na jejich místo vloží obsah atributu „copyNode“.

Posun kurzoru

`nextChar(shift:Bool = false)` Metoda, která změní aktuální pozici na následující znak, parametr nastavený na „true“ způsobí pokračování v označování.

`previousChar(shift:Bool = false)` Metoda, která změní aktuální pozici na předcházející znak, parametr nastavený na „true“ způsobí pokračování v označování.

`nextLine(shift:Bool = false)` Metoda, která změní aktuální pozici na následující řádek, parametr nastavený na „true“ způsobí pokračování v označování.

`previousLine(shift:Bool = false)` Metoda, která změní aktuální pozici na předcházející řádek, parametr nastavený na „true“ způsobí pokračování v označování.

`homeKey(shift:Bool = false)` Metoda, která implementuje přesun kurzoru na začátek řádku, parametr nastavený na „true“ způsobí pokračování v označování.

`endKey(shift:Bool = false)` Metoda, která implementuje přesun kurzoru na konec řádku, parametr nastavený na „true“ způsobí pokračování v označování.

Editace a formátování textu

`writeText(text:String)` Tato metoda vypíše zadaný text na aktuální pozici a provede nové zobrazení dokumentu.

`deleteKey()` Tato metoda odstraní následující znak případně celý aktuální výběr.

`backspaceKey()` Tato metoda posune kurzor o jeden znak doleva a odstraní následující znak. Pokud je vybráno více znaků, smaže celý výběr.

`enterKey()` Metoda, která v aktuálním místě vytvoří nový odstavec.

`coverRange(name:String)` Metoda, která obalí aktuální výběr formátovacím elementem daného jména.

6.2.3 LineContainer

Pomocí metod objektu container je možné vynutit nové zobrazení dokumentu.

`createLines()` Tato metoda provede nové parsování XML a celý dokument znovu zobrazí.

`reCreateLines(ind:Int)` Metoda, která nahradí pouze nevalidní řádky od odstavce zadaného indexem v parametru.

6.3 Problémy

6.3.1 Označování

Velmi mnoho času zabralo ladění označování. Kurzor je vždy nakreslen před znak, na který je kliknuto. Pro pohodlné označování většího textu myší směrem doprava tedy bylo nutné označovat o jeden znak navíc, jinak nebylo možné označit poslední znak v odstavci. U označování šipkami a klávesou Shift bylo toto chování zase nežádoucí. Kvůli označení posledního znaku v odstavci navíc musí být za posledním znakem ještě mezera, protože z ní získáme potřebné souřadnice. Označování prázdných odstavců komplikuje to, že Flash u nich nenastavuje zrcadlení událostí.

Zpočátku fungovalo označování myší pouze při pohybu přesně nad řádky. Potom se mi ale podařilo implementovat označování tak, že funguje i mimo řádky. Pokud se ukazatel vyskytuje na y-ové souřadnici v rozmezí řádku, je označen první nebo poslední atom podle toho, jestli je ukazatel nalevo nebo napravo od něj. V případě, že se na stejné y-ové souřadnici nevyskytuje žádný řádek, označování probíhá na prvním řádku směrem dolů od ukazatele myši.

6.3.2 Práce s XML

Během práce s novou třídou XML jazyka ActionScript 3.0 nebylo možné přeložit překladačem haXe základní konstrukce nutné pro práci s touto třídou. Nebylo možné jednoduše změnit hodnotu uzlu ani jej odstranit. Proto jsem zvolil pro práci s dokumentem původní třídy XMLDocument a XMLNode, které nezpůsobovaly žádné problémy. Pouze při nastavení parsování dokumentu na ignorování bílých znaků jsou ignorovány i obrázky, které neobsahují text. Tento problém byl vyřešen dvojím parsováním. Nejprve se parsuje bez ignorování bílých znaků a přidá se text do obrázkových uzlů. Poté se parsuje s ignorováním bílých znaků, protože jinak by bylo velmi těžké zjistit, které textové uzly opravdu patří do daného uzlu a které jsou pouze odsazením. Při ukládání dokumentu se text z obrázkových uzlů odstraní.

Náročné byly také optimalizace. Jednak XML stromu (spojování uzlů stejného typu), ale také nového zobrazení pouze části textu po malých editacích. To všechno ještě bylo zkomplikováno prací s obrázky – vytváření pomocných odstavců pro zarovnané obrázky, nastavování rozměrů a zobrazení dokumentu až po jejich načtení atd.

Při formátování textu obalováním elementem vznikají problémy u vnořených elementů. Během formátování uzlu se kontroluje pouze nejbližší předek (otec), tudíž může dojít ke zbytečnému vytváření formátovacího uzlu, když uzel již takto formátován je. Tento problém se zatím nepodařilo vyřešit.

6.4 Testování

Program byl testován pomocí ladícího programu „Projector“ dostupného na [2]. Tento program kromě zobrazení všech výjimek také umožňuje vytvoření spustitelného souboru s aplikací pro Windows.

Snažil jsem se otestovat většinu krajních případů. Příkladem můžou být posuny kurzoru, které nesmí způsobovat výjimky při překračování hranic textu. Dále obrázky se nesmí překrývat a způsobovat nesmyslné zalamování textu. Samozřejmostí je správná funkčnost všech tlačítek.

Testování výstupního dokumentu aplikace probíhalo pomocí validátoru W3C ([4]). Za typ dokumentu byl zvolen XHTML 1.0 Transitional, protože ve verzi Strict není možné použít atribut „align“ u obrázkových elementů. Výstup byl laděn tak, aby validátor nenahlásil žádnou chybu ani varování. Vzhled dokumentu v editoru byl porovnáván s jeho vzhledem ve webovém prohlížeči.

S laděním mi také pomohl vedoucí práce. Byl jsem upozorněn na problémy s označováním, na chybějící klávesové zkratky, nedostatky aplikačního rozhraní atd. Když jsem si nebyl jistý ohledně chování editoru, řídil jsem se příkladem nejlépe propracované Javascriptové komponenty – CKEditoru (viz kapitola 3.1.1).

Kapitola 7

Výsledky práce

Podařilo se mi implementovat zobrazení textu s odstavci, obrázky a XHTML tagy pro fyzické („b“, „i“, „u“, „s“) i logické („strong“, „em“) formátování. Označování textu funguje jak myší tak šipkami a klávesou Shift. Klávesy Home, End, Delete a Backspace mají standardní funkčnost nad kurzorem i výběrem textu. Klávesa Enter vytvoří nový odstavec, při použití Shift pouze nový řádek. Při kliknutí na tlačítka pro změnu formátování je výběr obalen elementem XHTML a zobrazen znovu. Cesta v XML stromu od aktuálního vybraného uzlu ke kořeni může být zobrazena v textovém poli. Při označování textu je také správně nastavena komponenta SelectBox na typ odstavce, který se dá snadno změnit.

Obrázek se označuje kliknutím myši nebo posunem kurzoru klávesami. Označený obrázek je možné vymazat, změnit tlačítky jeho zarovnání nebo přetažením myší umístit na jiné místo v textu. Tažením myši z dolního rohu obrázku se mění jeho velikost, klávesa Shift ovlivňuje to, zda bude zachován původní poměr šířky a výšky obrázku.

Jsou implementovány standardní klávesové zkratky pro kopírování (Ctrl+C, Ctrl+Insert), vyjmutí (Ctrl+X, Shift+Delete) a vkládání textu (Ctrl+V, Shift+Insert). Tyto operace zachovávají formátování textu. Pokud dokument překračuje zadanou výšku, je ořezán a je možné jej posouvat pomocí kolečka myši. V ukázce použití editoru jsou využita tlačítka a SelectBox, které jsem získal od vedoucího práce, a posuvník, který pochází z projektu „haXe minimalcomps“ ([9]).

Výsledná komponenta je jednoduše použitelná, může být součástí větších komponent. Uživatel si může zvolit, které editační funkce chce mít zobrazeny. Je možné ji použít i jen jako pole pro zobrazení a editaci textu a obrázků. Výstupem programu je validní XHTML 1.0 Transitional. Jedním z cílů byla i snadná rozšiřitelnost, o které se více dozvíte v následující kapitole.

7.1 Rozšiřitelnost

Program je možné jednoduše rozšířit zásahem do zdrojového kódu. K tomu je důležitá znalost aplikačního rozhraní, jehož popis je v kapitole 6.2.

Příkladem rozšíření může být přidání dalšího tlačítka pro obalení elementu. Nejprve se vytvoří nové tlačítko ve třídě Example. Může být libovolného typu, ale musí se předat třídě Editor metodou `setFormattingButtonType`, aby formátování fungovalo. Potom se události kliknutí na tlačítko přiřadí metoda, která zavolá funkci `range.coverRange` a předá jí název elementu. Metoda třídy Range se postará o obalení vybraného textu daným element i o nové zobrazení dokumentu.

Dalším rozšířením by mohla být změna zarovnání odstavce na blokové. Tuto metodu by ale bylo vhodné pro zjednodušení napsat do zdrojového kódu třídy Range. Vypadala by podobně jako metoda `changeParagraphType`, jediným rozdílem by bylo, že místo názvu nastavuje atribut `align`. Tato metoda by se navázala na nové tlačítko. Aby editor správně zobrazil zarovnání takto označených odstavců, musí se ještě editovat metoda `parseHTML` třídy MyXMLDocument, kde se do volání funkce `parseNode` předá upravený objekt ElementFormat.

Jak je vidět z příkladů, některá rozšíření je možné realizovat bez zásahu do kódu jednotlivých tříd editoru. Složitější rozšíření, která mění způsob zobrazení dokumentu, se ale bez zásahu do těchto tříd neobejdou.

7.2 Možnosti dalšího rozvoje komponenty

Možností rozvoje této komponenty je mnoho. V editoru chybí možnost měnit velikost a typ písma a mezery mezi řádky a odstavci. Nyní je možné tyto atributy změnit pouze přepsáním hodnot konstant v jednotlivých třídách. Dále by měla být přidána možnost vytvářet číslované a nečíslované seznamy, měnit zarovnání odstavců a přidávat obrázky jiným způsobem než editací vstupního dokumentu. Některé editory umožňují i náhled zdrojového kódu zpracovávaného dokumentu a jeho editaci. Důležitým prvkem, který zatím v komponentě chybí, je možnost vytvářet odkazy. Příkladem dalších pokročilých funkcí může být tvorba tabulek a formulářů nebo práce s kaskádovými styly.

Kapitola 8

Závěr

Cílem bakalářské práce bylo vytvořit open-source editor jazyka XHTML v prostředí Flash. V kapitole 3.2 jsou popsány jednotlivé již existující nástroje, které mají stejný cíl, ale z nichž zároveň každý disponuje i velkými nevýhodami – většinou nepracují správně s obrázky nebo jsou komerčního charakteru. Proto je účelem práce vytvořit novou svobodnou komponentu, která v této oblasti chybí.

Abych mohl tento nástroj implementovat, bylo nutné si nastudovat skriptovací jazyk ActionScript 3.0 ([7]) a seznámit se s jeho novou knihovnou Flash Text Engine ([11]). Pro implementaci byl zvolen open-source jazyk haXe ([3]), který sice nemá vývojové prostředí, ale umožňuje snadno vytvářet Flash komponenty a má velmi podobnou syntaxi jako jazyk ActionScript. Studium zmíněných materiálů mi bylo přínosem, protože pro mě byla tato oblast úplně nová.

Výsledkem práce neměla být komponenta, která by zahrnovala všechny potřebné funkce, ale pouze snadno rozšiřitelný základ editoru. Z ukázkové aplikace je patrné, že cíle bylo dosaženo. Editor umožňuje základní práci s textem, jeho formátování a především manipulaci s obrázky (zarovnávání, přesun a změnu velikosti). Tuto funkcionalitu řada standardních editorů postrádá. Proto může výsledná komponenta nahradit tyto nástroje a najít tak uplatnění v praxi.

Editor je navíc možné snadno rozšířit o další funkce, např. změna zarovnání odstavců nebo vytváření číslovaných a nečíslovaných seznamů. Řešení několika jednodušších rozšíření je popsáno v kapitole 7.1.

Literatura

[1] ActionScript 3.0 Reference for the Adobe Flash Platform [online]. http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/, 2000-04-29 [cit. 2011-04-06].

[2] Adobe Flash Player Support Center [online]. http://www.adobe.com/support/flashplayer/downloads.html, 2009-07-14 [cit. 2011-04-11].

[3] haXe Documentation [online]. http://haxe.org/doc, 2010 [cit. 2011-04-06].

[4] Markup Validation Service [online]. http://validator.w3.org/, 2010 [cit. 2011-05-06].

[5] Brossier, V.: Flash Text Engine [online]. http://www.developria.com/2009/03/flash-text-engine.html, 2009-03-19 [cit. 2011-04-20].

[6] Hégaret, P. L.; Whitmer, R.; Wood, L.: Document Object Model (DOM) [online]. http://www.w3.org/DOM/, 2005-01-19 [cit. 2011-04-06].

[7] Hozík, M.: Úvod do ActionScriptu 3.0 [online]. http://flash.jakpsatweb.cz/actionscript-3/, 2011 [cit. 2011-04-11].

[8] Němeček, L.: *WYSIWYG editor*. Diplomová práce, FEL ČVUT v Praze, 2010.

[9] Peters, K.: MinimalComps [online]. http://www.minimalcomps.com/, 2008 [cit. 2011-05-04].

[10] Sharpe, P.; Apparao, V.; Wood, L.: Document Object Model Range [online]. http://www.w3.org/TR/DOM-Level-2-Traversal-Range/ranges.html, 2000-11-13 [cit. 2011-04-06].

[11] Taylor, P.: TINYTLF [online]. http://guyinthechair.com/tag/tinytlf/, 2010-06-03 [cit. 2011-04-16].

Příloha A

Obsah CD

- Soubor „readme.txt“ s návodem na překlad a spuštění aplikace,
- Adresář s přeloženou komponentou (`run/`),
- Adresář s dokumentací vygenerovanou pomocí programu HaxeDoc (`doc/`),
- Zdrojové kódy programu (`src/`),
- Text práce v elektronické podobě (adresář `text/`),
- Adresář se zdrojovým textem práce (`text/src`),
- Adresář s použitými knihovnami (`lib/`)