VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

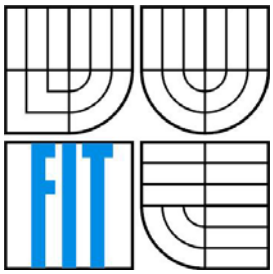
AUTOMATICKÝ GENERÁTOR KOMIKSŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. PAVEL HOLEC
AUTHOR

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÝ GENERÁTOR KOMIKSŮ

AUTOMATIC COMICS GENERATOR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL HOLEC

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ADAM HEROUT, Ph.D.

BRNO 2007

Zadání diplomové práce

Řešitel:	**Holec Pavel, Bc.**
Obor:	Počítačová grafika a multimédia
Téma:	**Automatický generátor komiksů**
Kategorie:	Počítačová grafika

Pokyny:

1) Seznamte se s problematikou vytváření komiksů, vyhledejte a popište existující nástroje.
2) Navrhněte a popište vhodné filtry zpracování obrazu, produkující obrázky vhodné do komiksu.
3) Navrhněte a vytvořte program, který by ze zadaného textu automaticky generoval komiks.
4) Demonstrujte funkčnost programu na několika vygenerovaných anglických a českých komiksech.
5) Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek pro prezentování projektu.

Literatura:

dle pokynů vedoucího

Část požadovaná pro obhajobu SP:

body 1.-2., dílčí práce směřující k naplnění bodu 3.

Vedoucí:	**Herout Adam, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání:	28. února 2006
Datum odevzdání:	22. května 2007

Vedoucí ústavu: **doc. Dr. Ing. Pavel Zemčík**

Licenční smlouva

Licenční smlouva je uložena v archivu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Obsahem této práce je implementace programu, který umožňuje automatické generování komiksové stránky na základě vstupního komiksového textu. Zabývá se dále technikami zpracování obrazu a implementací grafických filtrů, které z fotografií produkují kreslený či jinak umělecky pojatý obrázek.

Klíčová slova

komiks, zpracování obrazu, uživatelské rozhraní, .NET Framework, GDI+, Google, HTML

Abstract

The subject of this thesis is an implementation of program that enables automatic generation of a comics page from input comics text. It also discusses various techniques of image processing and implementation of graphical filters that can transform a photograph into a drawing or another artistic picture.

Keywords

comics, image processing, user interface, .NET Framework, GDI+, Google, HTML

Citace

Pavel Holec: Automatický generátor komiksů, diplomová práce, Brno, FIT VUT v Brně, 2007

Automatický generátor komiksů

Prohlášení

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

.....
Pavel Holec
22. 5. 2007

Poděkování

Rád bych poděkoval vedoucímu práce za pomoc při její tvorbě, panu Ing. Jaroslavu Rábovi za pomoc s nastavením testovacího serveru, panu Iwama Kazu za pomoc s japonským jazykem, přátelům za podporu a pomoc s vytvořením ukázkových komiksů.

Obsah

1 Úvod

V dnešní době jsou již běžně využívány počítačové programy jako nástroje k vytvoření nejrůznějších uměleckých děl. U mnoha z těchto programů v oblasti grafiky, hudby a jiných multimedií, je již mnohdy výsledné dílo více ovlivněno vnitřními algoritmy a postupy aplikace než samotným tvůrcem. Ten již nemá absolutní kontrolu nad výsledkem; stává se spíše jakýmsi zadavatelem úlohy a dozorcem nad jejím vykonáním. Nezřídka je tvůrce sám překvapen výsledkem a popravdě řečeno, kdyby jej měl napodobit pouze vlastní zručností bez pomoci počítače, možná by nedosáhl ani z části tak kvalitního výsledku.

Díky tomu je v dnešní době zpřístupněna tvorba multimedií i uživatelům, kteří nemají v daném oboru potřebné teoretické znalosti nebo trénovanou manuální zručnost. My jsme v této práci nahlédli do tradičního postupu tvorby komiksů. Pokusili jsme se tento proces plně automatizovat a zjistit, zda je možné implementovat program, který by umožnil libovolnému uživateli tvorbu komiksové stránky.

Cílem implementovaného programu je tedy tvorba komiksové stránky. Po uživateli se požaduje pouze nastavení požadovaného tématického vzhledu komiksu a vstupní text (například příběh, oznámení, vtip), který je analyzován a vhodně rozčleněn do popisků komiksových políček. Jako pozadí těchto políček slouží obrázky, upravené pomocí navrhnutých grafických filtrů tak, aby barevně a tématicky ladily se zbytkem komiksové stránky. Obrázky se získají pomocí zvolené online vyhledávací služby (například ***Google images***[1]), díky které se dá vyhledat náhodný obrázek či fotografie pod vhodným klíčovým slovem. Toto klíčové slovo je produktem analýzy vstupního komiksového textu za pomoci vyhledávací služby v online slovníku.

Díky pestrému množství obrázků, které vyhledávací služby indexují a díky jejich variabilitě je hlavním využitím programu možnost generování humorných přání, pozvánek, vtipů nebo příběhů s pozměněným významem. Jelikož v dnešní době neexistuje vyhledávací služba, která by indexovala takové množství obrázků (stamilióny) s kvalitními a přesnými výsledky, nemůže naše implementace zajistit kvalitnější výsledky vyhledání obrázků a je tak plně závislá na budoucím zlepšení těchto vyhledávacích služeb.

[1] http://images.google.com

1.1 Souhrn kapitol

Na konci úvodní kapitoly zmíníme některé často použité pojmy z prostředí tradiční tvorby komiksů, grafických filtrů a implementačního prostředí, které budeme v práci dále používat.

Ve druhé kapitole se zaměříme na již dostupné nástroje pro automatickou tvorbu komiksů a na způsob, jakým komiks vytvářejí, spolu s ukázkou jejich výstupu.

Třetí kapitola přiblíží vizuální stránku a strukturu tradičního komiksu[1]. Jsou zde popsány jednotlivé grafické elementy, ze kterých se komiks skládá a také základní výtvarné styly komiksových obrázků, které vyplňují komiksová políčka.

Představením prostředků a algoritmů, zvolených k implementaci programu, se zabývá čtvrtá kapitola. Zmiňuje programovací prostředí ***.NET Framework***, grafické rozhraní ***GDI+*** a rozhraní pro komunikaci s internetovými stránkami pomocí internetového protokolu ***HTTP***. Popisuje dále možnosti vytvoření online klientského rozhraní pomocí dynamických stránek ve formátu ***PHP***. Dále se v této kapitole budeme věnovat kombinacím v praxi již existujících filtrů, které po aplikaci na fotografii vytváří její zajímavé kreslené variace, vhodné pro použití do komiksu. Poslouží nám jako inspirace při implementaci našich vlastních algoritmů pro zpracování obrazu.

Pátá kapitola se již zabývá samotnou implementací programu. Popisuje schéma běhu programu, algoritmy pro vytvoření a vykreslení stránky, algoritmus pro volbu vhodného klíčového slova pro nalezení odpovídajícího obrázku a algoritmy pro aplikaci obrazových filtrů. Dále je v této kapitole popsána implementace grafického uživatelského rozhraní a ovládacích prvků.

Předposlední kapitola se zabývá zhodnocením naší implementace. Uvádíme zde výsledky z oblasti dělení textu, rychlosti nalezení klíčových slov a filtrace obrazu.

Závěrečná kapitola je věnována shrnutí diplomové práce a možnostem budoucího rozšíření a vylepšení implementovaného programu a dodatečného online klientského rozhraní.

[1] http://www.balloontales.com/

1.2 Návaznost na semestrální projekt

Semestrální projekt obsahoval funkční implementaci programu, který by již zadání práce vyhovoval. Zkoumali jsme však dále možnosti filtrace obrazu pomocí skriptování pokročilých grafických editorů. Velikých změn doznalo grafické uživatelské rozhraní, které nyní umožňuje interaktivní změnu komiksových elementů a nabízí daleko více funkcí pro nastavení vzhledu komiksu a jeho manipulaci. Aplikace byla tímto více zpřístupněna širší skupině uživatelů.

Nedílnou součástí byla optimalizace celého programu pro zajištění vyšší rychlosti generování komiksu. Také algoritmy pro vyhledání klíčových slov a nalezení obrázku byly od základu změněny a vylepšeny. Posledním bodem, navrženým v semestrální práci, bylo vytvoření dodatečného zkušebního internetového prostředí naší aplikace.

1.3 Základní pojmy

Komiks (comics) – forma výtvarného umění[1]. Sestává z obrázků, které obsahují text ve formě textových bublin a popisků. Text a obraz zde mají pro vyprávění příběhu stejnou váhu. Původní funkcí komiksu bylo pouze pobavit čtenáře krátkými vtipnými příběhy (stripy), zanedlouho se však vyvinul v literární celek s mnoha žánry.

Strip – velmi krátký humorný komiks, většinou se vejde do pár políček. Často bývá publikován v novinách či na internetu na pokračování.

Popisek (caption) – textová bublina pravoúhlého tvaru, ve které je umístěn popis události.

Políčko (frame) – útvar převážně pravoúhlého tvaru obsahující obrázek. Může obsahovat textové bubliny či popisky.

Pozadí (background) – nejspodnější grafická vrstva komiksové stránky. Většinou se skládá pouze z barvy. Pokud obsahuje obrázek, je pro přehlednost většinou vykreslen pouze v části, kde se nachází nadpis komiksu.

Žlábek (gutter) – prázdné místo mezi panely, které odkrývá pozadí.

Nadpis (title) – nadpis komiksu, uvedený na titulní nebo úvodní straně nahoře, většinou zabírá celou šířku stránky a je psán bohatým fontem.

Titulky (credits) – informace o autorovi, datum vydání a další podobné údaje o původu komiksu.

Rozvržení stránky (layout) – rozložení a umístění výše zmíněných komiksových elementů na ploše komiksové stránky.

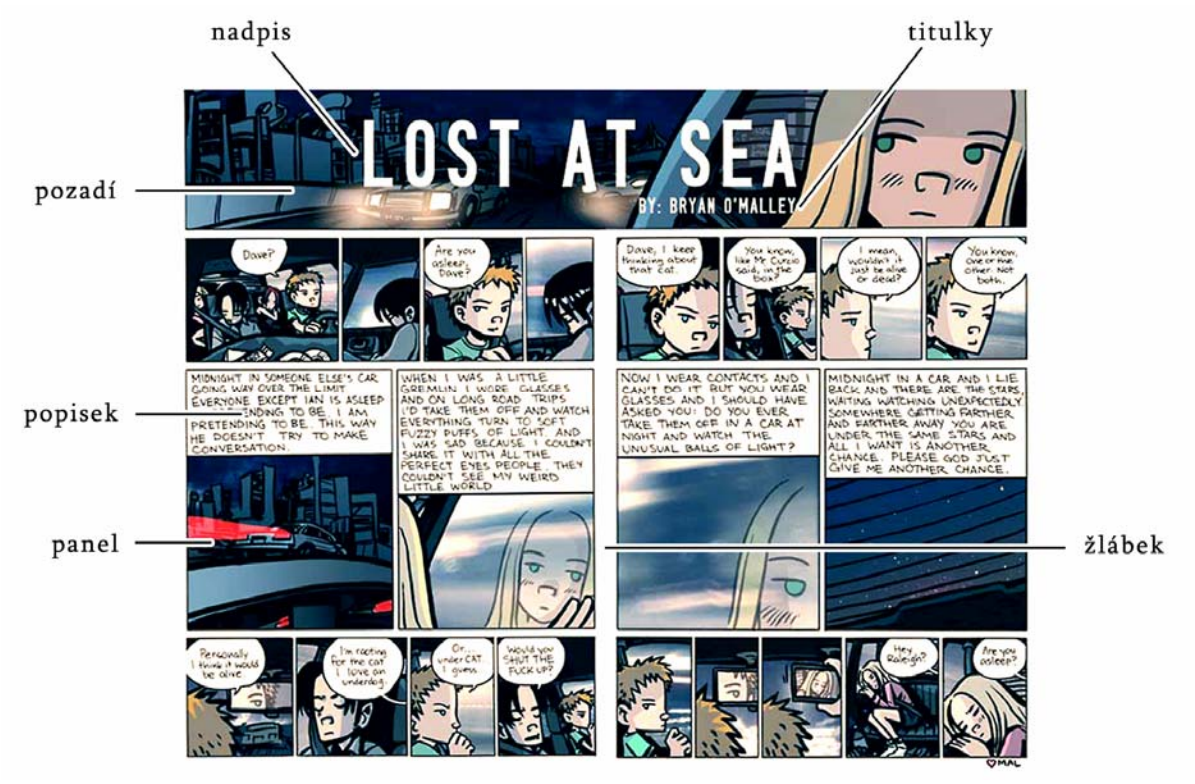
Posterizace (posterization) – redukce počtu barev v obraze pomocí různých technik prahování.

[1] Wikipedia, the free encyclopedia: *Comics*. http://en.wikipedia.org/wiki/Comics [duben, 2007]

Polotónování (halftoning) – tiskařská technika reprezentace obrazu vzorkem malých teček o různém průměru. Původně byla využívána jako způsob tisku obrázků s různým počtem barev na zařízení, které obsahovalo řádově menší počet barev. Její digitální obdoba umožňuje ze vstupního obrazu vytvořit obraz, zdánlivě vytisknutý na tiskárně v nižší kvalitě bodového rozlišení.

Pero (pen) – perem pro naše účely rozumíme grafický objekt v námi zvoleném programovacím prostředí; nese informace o typu, tvaru a barvě obtažení vektorového objektu.

Štětec (brush) – obdobně jako u pera se jedná o grafický objekt. Nese informace o typu, tvaru a barvě výplně vektorové objektu.



(Obrázek 1) rozložení základních grafických elementů v komiksu[1]

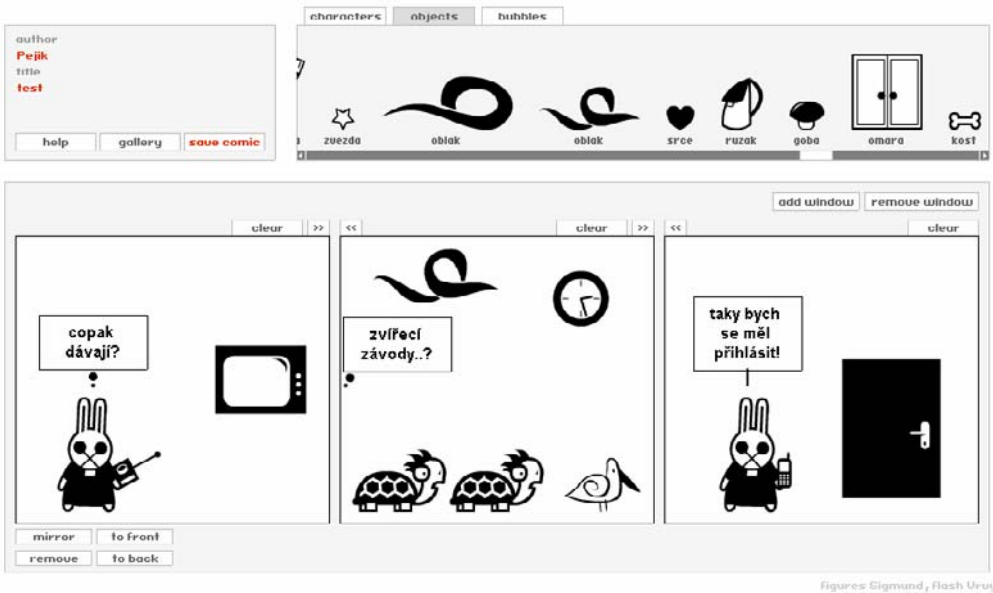
[1] O' Malley B.L.: *Lost at Sea*. Oni Press, 2006

2 Existující nástroje

Po důkladném prozkoumání online zdrojů jsme zjistili, že nástrojů pro automatickou tvorbu komiksů není mnoho. Většinou se jedná o programy, běžící na webovém rozhraní, a umožňují snadno vytvořit krátký strip spolu s jeho názvem a jménem autora. Uživatel rozmístí do každého políčka předem definované objekty (postavy, předměty nebo pozadí) a dodá textové bubliny s požadovaným textem. Někdy bývají pozadí i postavy předem umístěné a lze pouze doplnit text do textových bublin. Vytváření komiksu je poměrně jednoduché a rychlé, ovšem nutnost použití předdefinovaných obrázků omezuje vizuální pestrost výsledku. Nejzajímavější generátory komiksů představíme a popíšeme jejich hlavní výhody a prvky, které nás inspirovaly.

Strip generator

Nástroj ***Strip generator***[1] je propracovaný generátor krátkých komiksových stripů, vytvořený v programovacím prostředí ***Flash***. Používá předem definované grafické objekty, které se vkládají do komiksových políček spolu s textovými bublinami a popisky. Počet políček závisí na uživateli; dají se snadno přidávat nová, ovšem pouze v horizontálním směru. Celý postup tvorby komiksu je manuální. Předdefinované objekty jsou černobílé, výrazně umělecky stylizované do podoby piktogramů. Po vytvoření komiksu je možné jej zaslat na zvolenou elektronickou adresu. Program má dobrou podporu co se týče galerie a hodnocení hotových komiksů a registrace nových uživatelů. Počet tímto nástrojem dosud vytvořených komiksů je již kolem dvaceti tisíc.

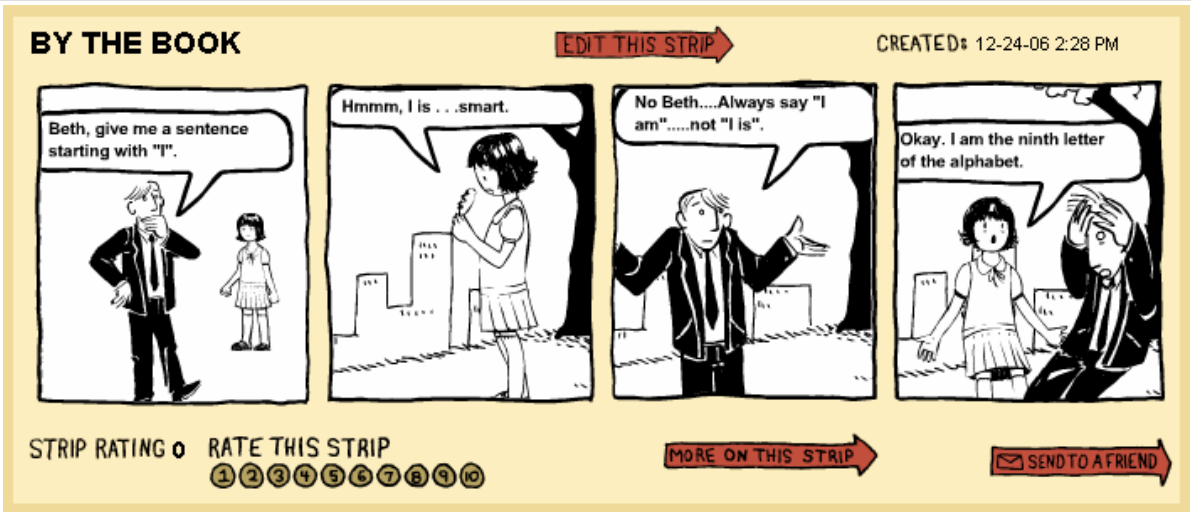


(Obrázek 2) grafický výstup programu ***Strip generator***

[1] Thirdframestudios: *Strip generator*. online aplikace, verze 1.0.2. http://stripgenerator.com

Commix

Commix[1] je další zajímavý generátor komiksů, funkčně podobný programu ***Strip generator***. Předdefinované sady objektů zde pocházejí od různých grafiků; výsledné komiksy jsou tedy různorodější. Textové bubliny i rozvržení stránky mají předem určené sady velikostí. I zde je použito rozhraní aplikace v prostředí ***Flash*** a rovněž je možné hodnotit již vytvořené komiksy.



(Obrázek 3) grafický výstup programu ***Commix***

Ostatní nalezené nástroje pro generování komiksů nenabízely nic nového. Další zajímavé generátory komiksů již jen výčtem:

Build Your Own Meat[2] – umožňuje vytváření stripů, které napodobují známý strip ***Red Meat***[3].
Comic creator[4] – nástroj pro vytváření komiksů ze světa Garfieldových příběhů.
Witty comics[5] – nabízí pestré množství kreslených pozadí a postav.

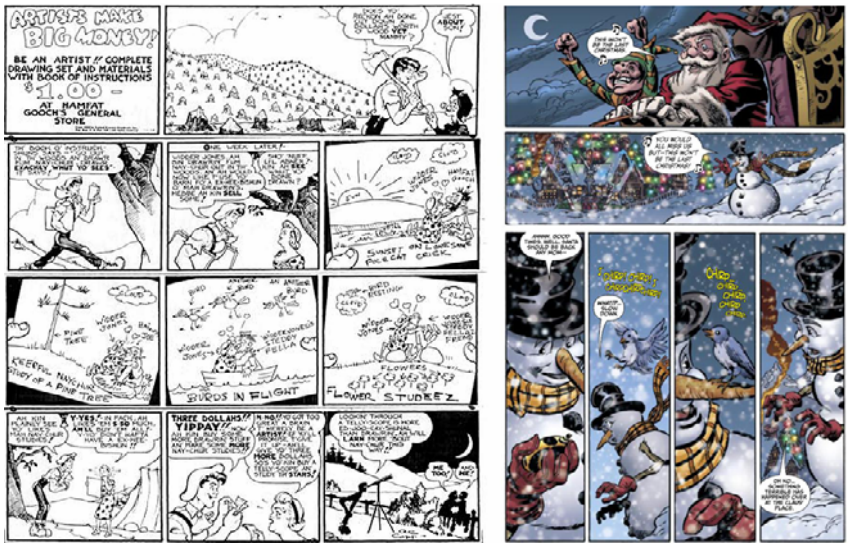
Nalezené nástroje jsou s naší aplikací poměrně rozdílné. Jejich automatizace zpracování spočívá pouze v jednoduchém manuálním vytvoření a umístění komiksových prvků, zatímco naše aplikace si kladla za cíl plně automatické vytvoření komiksu bez dalšího zásahu uživatele (přestože disponuje bohatými možnostmi manuálního nastavení). Nenabízejí nastavení barevnosti objektů (často generují pouze černobílé obrázky) ani žádnou formu zpracování obrazu; disponují pouze sadou předdefinovaných objektů bez možnosti vložit do komiksu vlastní objekty nebo obrázky.

[1] Willmon J.: *Commix*. online aplikace, verze z roku 2006. http://www.com-mix.org
[2] http://monkeydyne.com/rmcs/buildmeat.html
[3] http://www.redmeat.com/redmeat/
[4] http://www.nhlbi.nih.gov/health/public/sleep/starslp/missionz/comic.htm
[5] http://www.wittycomics.com/make-comic.php

3 Grafické ztvárnění komiksu

3.1 Typický vzhled komiksu

Základní stavební prvky komiksu jsou stejné již mnoho desítek let. První komiksy vznikaly sice již v devatenáctém století, ale podobu, kterou je dnes komiks charakterizován, získal teprve na počátku dvacátého století. Vzhled komiksu se řídí ustálenými pravidly, proto si komiks uchovává stálou vizáž, kterou se budeme snažit napodobit.



(Obrázek 4) šedesát let starý, ručně kreslený komiks[1] a novodobý komiks[2], vytvořený pomocí počítače

Častou formou komiksu jsou stripy; krátké příběhy o několika málo políčkách. Převládají stripy o třech políčkách v horizontálním rozvržení. Platí v nich stejná pravidla jako ve větších komiksech s tím rozdílem, že nadpis a titulky jsou malým písmem začleněny do prostoru stripu.

(Obrázek 5) typický komiksový strip[3] o třech políčkách

[1] Frazetta F.: *Al Capp's Li'l Abner: The Frazetta Years, Volume 1*. Dark Horse, 2003

[2] Duggan G., Posehn B.: *The last Christmas*. Image Comics, 2006

[3] Davis J.: *Garfield Classics: Vol 14*. Ravette Publishing Ltd, 2004

3.2 Grafické prvky komiksu

Pro napodobení komiksu je třeba komiks analyzovat z grafického hlediska a rozdělit jej na jednotlivé komiksové elementy, ze kterých se skládá. To nám umožní jej následně z těchto grafických prvků zpětně vytvořit. Budeme se snažit je napodobit pomocí námi implementovaných grafických prvků. Z mnoha ukázek komiksů jsme vybrali následující společné elementy.

Pozadí

Za všemi nadpisy, titulky i panely je umístěno pozadí, jehož hlavní funkce spočívá v navození atmosféry komiksu. Běžně se například v novinách užívá neutrální bílé pozadí (jinými slovy žádné pozadí), zatímco komiksy s napjatou atmosférou využijí spíše tmavé až černé pozadí.

Aby pozadí nevypadalo obyčejně, používá se někdy místo jednolité barvy jemný barevný přechod. Pozadí bývá doplněno někdy nenápadnou, někdy naopak výraznou kresbou. Ta se vyskytuje převážně v místech, kde se nachází nadpis komiksu, aby nerušila čitelnost zbytku komiksu. Výjimkou jsou kresby, které přímo zasahují do vyprávění a slouží jako úvodní komiksové políčko pro uvedení do děje komiksu.

(Obrázek 6) pozadí komiksu – bílé[1], černé[2], barevný přechod[3], výrazné[4] (zasahující do vyprávění)

[1] McCay W.: *Little Sammy Sneeze*. Checker Book Publishing Group, 2004

[2] Fawstin B.: *Table for one*. Mainspring Comics, 2004

[3] Parker S., Alexander J.: *Across the Pond Presents #2*. Across the Pond Studios, LLC, 2004

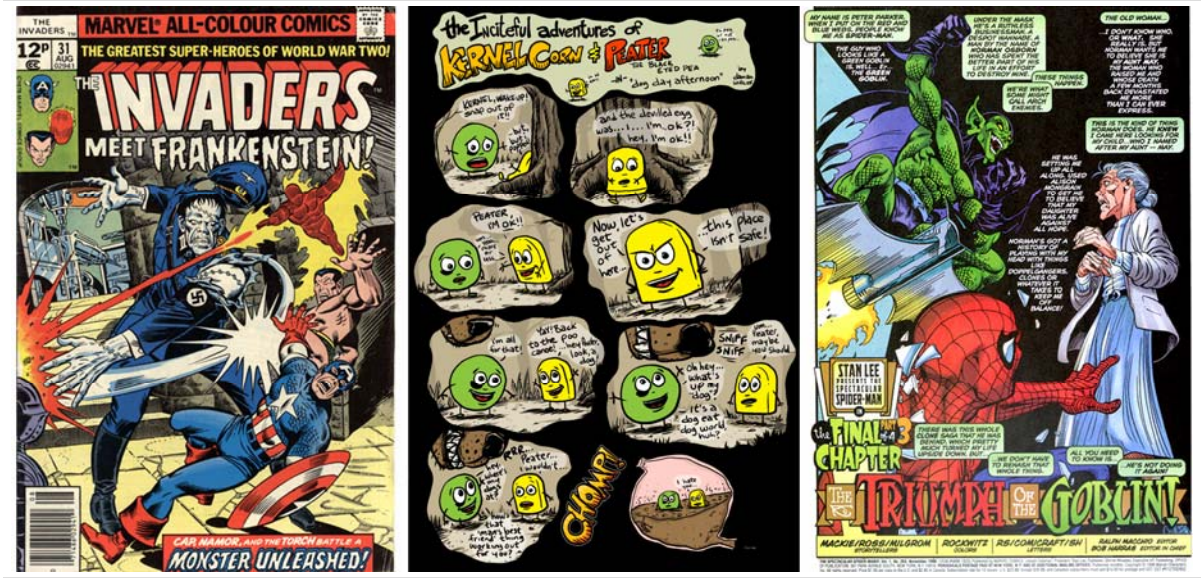
[4] http://www.applegeeks.com/comics/

Nadpis

Nadpis komiksu se používá buď na titulní, nebo první stránce komiksu; je psán bohatým fontem a značnou velikostí písma. Bývá ozvláštněn vizuálními efekty, jako jsou stíny nebo trojrozměrné efekty. Často je svým stylem tématicky spjat s náladou komiksu a prezentuje tím jeho styl. Většinou je mu vyhrazena samostatná úvodní stránka spolu s vyobrazením hlavních hrdinů komiksu.

V komiksech, které jsou rozvrženy pouze na jednu stránku, se nachází ve většině případů v horní části, kde má pro své prezentační účely vyhrazenou poměrně značnou část prostoru. Jiné umístění bývá použito jen výjimečně.

V krátkých stripech z novin bývá takto bohatý název komiksu vynechán a lze zde pouze najít název společný pro celou sérii a jméno autora, psané běžným fontem. Často je pak také umístěn nenápadně mezi komiksovými políčky.



(Obrázek 7) nadpisy – na samostatné úvodní stránce[1], v horní části[2], méně tradičně ve spodní části[3]

[1] Marvel Comics: *Invaders meet Frankenstein #31*, Marvel Comics, 1978

[2] http://www.dorkboycomics.com/

[3] Mackie H.: *Spectacular Spider-Man #263*, Marvel Comics, 1998

Políčka

Komiksová políčka bývají téměř vždy pravoúhlá s úzkým černým orámováním. Někdy bývají ozvláštněna netypickým tvarem nebo prolnutím s jinými políčky či s pozadím. Jejich rozmístění se dá reprezentovat pomyslnou mřížkou, která je však většinou pro odstranění stereotypu v horizontálním směru nepravidelná. Starší komiksy se vyznačovaly téměř striktní pravidelností této mřížky, dnes je trendem používat různé velikosti políček, z nichž některé mohou svoji výškou přesahovat i několik řádků mřížky. Někdy je proto potřeba doplnit kresbu o šipky, které určují směr čtení políček, jinak by se čtenář mohl v komiksu snadno ztratit.

(Obrázek 8) typy políček (shora dolů) – klasická[1], kombinovaná[2], umělecká[3]

[1] Duggan G., Posehn B.: *The last Christmas*. Image Comics, 2006

[2] http://jamesblond.transplantcomics.com/

[3] Anderson E. A., Tembley M.: *Sam Noir samurai detective*, Image Comics, 2007

Titulky

Titulky se nejčastěji nachází buď poblíž nadpisu, v našem případě tedy v horní části stránky, nebo zcela ve spodní části. Obsahují informace o tvůrcích komiksu, nakladatelství, a jiné dodatečné údaje. Jsou nejčastěji psány stejným fontem jako text celého komiksu, pouze menší velikostí, a nacházejí se volně na pozadí.

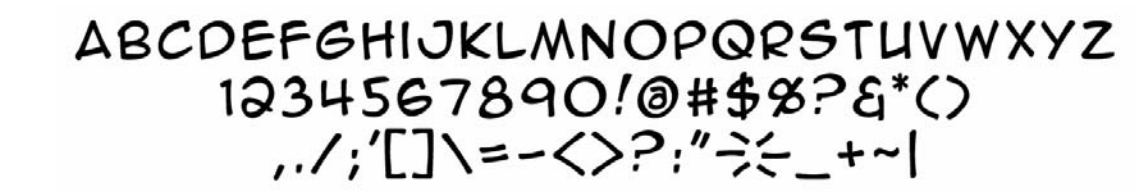
Popisek

Jelikož nám forma textu umožňuje pouze popis události či vyprávění a ne dialogy konkrétních postav, nebudeme se zabývat klasickými textovými bublinami, ale pouze komiksovými popisky. Ty jsou umístěny v rámečku o pravoúhlém tvaru, zpravidla ohraničeném jednoduchou černou čarou. Kvůli čitelnosti textu je voleno jednobarevné pozadí, kontrastující s barvou písma textu. Občas se pro pozadí používá i jemný barevný přechod. Popisek je většinou umístěn blízko okrajů políčka, může přes něj i přečnívat; někdy také vyplňuje celou jeho šířku.

(Obrázek 9) popisky – u okraje[1], volně[1] (s přesahem), přes šířku políčka[2], s barevným přechodem[3], kreslené[2]

Text

Text v komiksu není tak objemný jako v klasické literatuře. V komiksovém popisku se nenachází více než pár vět, proto může být psán větším a bohatším písmem než klasické knihy. Font komiksového textu vychází z původní podoby ručně kresleného textu používaného v humorném stripu. Tento styl se uchoval dodnes i pro zcela odlišné žánry než je humorný komiks. Existuje již celá řada fontů, dostupných v digitální podobě. Používají se dnes při tvorbě komiksů pro úsporu práce, například při překladu komiksu z cizího jazyka.



(Obrázek 10) typický font pro tvorbu komiksů – ***Anime Ace***

[1] Lee S., Roy T.: *Spider-Man: Saga of the Sandman*. Marvel Comics, 2007

[2] Straczynski J.M., Youngquist J.: *Supreme Power*. Marvel Comics, 2005

[3] Mackie H.: *Spectacular Spider-Man #263*, Marvel Comics, 1998

Obrázky v políčkách

Obrázky, které vyplňují komiksové políčko, jsou nejvíce variabilním prvkem celého komiksu a často již podle jejich stylizace lze uhodnout jejich autora. Zde nelze než vybrat pouze nejzajímavější typy stylizace a pokusit se je rozdělit do několika hlavních směrů, které ovlivní výběr a návrh grafických filtrů, potřebných pro převod fotografie na kresbu.

Starý tištěný styl

Reprezentuje barevné obrázky z počátku devatenáctého století. Díky nekvalitnímu tisku mají výrazné tónovací vzory, obsahují jen pár základních barev bez stínování a barevných přechodů.

Styl noir

Černobílé ztvárnění s dominantní černou barvou a extrémně vysokým kontrastem, prakticky bez stínování. Občas je doplněno o barvu navíc, která je užívána zřídka a určuje téma příběhu.

Styl malby

Obrázky se podobají malbě, mají tedy blíže k realismu než předchozí styly.

Fotky s úpravou barev

Styl, ve kterém se užívají místo kreslených obrázků fotky se změnou barev a vyšším kontrastem. Používá se například pro tvorbu komiksu podle filmové předlohy. Tímto stylem se také vyznačují komiksy, vytvořené s pomocí našeho programu.

(Obrázek 11) některé výrazné styly komiksových obrázků – starý komiks[1], noir[2], malba[3], stylizovaná fotka[4]

[1] Lane G., Guillaumat F.: *La loi de Bitur-Camember*. 2002

[2] Miller F.: Sin City: *That Yellow Bastard Bk. 4*. Dark Horse Comics, 2005

[3] http://www.jentong.com/comics.html

[4] http://suicidegirls.com/

4 Prostředky pro implementaci programu

4.1 Programovací platforma .NET Framework

Pro implementaci programu jsme (nejen z důvodu pokročilého zpracování obrazu) vypustili možnost použití některého internetového programovacího prostředí, jako je ***Flash*** či ***PHP***. To by nám ani s podporou některých modulů na podporu generování grafických elementů nemohlo poskytnout zdaleka tolik možností pro finální kvalitu generovaného komiksu. Zvolili jsme tedy programovací prostředí ***.NET Framework*** spolu s programovacím jazykem ***C#***, především pro bohaté možnosti vykreslování pomocí grafického aplikačního rozhraní ***GDI+***, dobré možnosti vytvoření profesionálního vzhledu grafického uživatelského rozhraní a také pro umožnění bezproblémové komunikace s internetovým protokolem ***HTTP***.

Implementace programu byla konkrétně provedena v programovacím prostředí ***Microsoft Visual Studio 2005***, provozovaném pod školní licencí. Použili jsme pouze vnitřně dostupné nástroje tohoto rozhraní; veškeré další technické prostředky (nestandardní elementy grafického uživatelského rozhraní) jsme implementovali sami. Přitom jsme často čerpali z ***MSDN Library***[1]. Ke spuštění programu je tedy potřeba mít nainstalovanou platformu ***.NET Framework*** verze 2.0 nebo vyšší.

4.1.1 GDI+

Grafické aplikační rozhraní ***GDI+*** oproti své starší verzi (***GDI***) zjednodušuje implementaci vykreslovacích úloh, bohužel však na úkor rychlosti běhu aplikace. Je objektově orientované a dobře zapadá do konceptu jazyka ***C#***. Nabízí nové možnosti jako je vykreslování cest (objekt typu ***Path***), typografické nástroje, podporu pro mnoho grafických souborových formátů (například ***Bmp***, ***Jpeg***, ***Png***), transformace obrazu a mnoho dalších funkcí, které naše implementace využívá.

Rozhraní pracuje nad objektem typu ***Graphics***, které si můžeme představit jako plátno, na které se vykreslují grafické elementy. Těchto objektů využíváme v aplikaci několik; pro námi definované vykreslení komponent grafického uživatelského rozhraní, dále pro vykreslování náhledového okna, a v neposlední řadě především pro samotné vykreslení komiksu. Grafickému objektu je potřeba nastavit kvalitu vykreslení v různých kategoriích (pro vykreslení textu, bitmap, křivek, atd.), jinak má nastavené standardní hodnoty, které nemusí vždy vyhovovat. Toto nastavení se velmi projevuje na rychlosti v aplikacích, ve kterých se při interaktivních operacích periodicky překresluje obraz, což je právě náš případ.

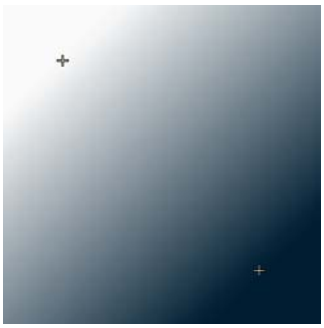
[1] http://msdn2.microsoft.com/en-us/library/default.aspx

V následujících odstavcích uvedeme některé, námi nejčastěji využívané, metody třídy ***Graphics***.

DrawString – metoda pro vykreslení textu o zvoleném fontu a velikosti. Využíváme ji k vykreslení nadpisu, titulků a samotného komiksového textu. Styl výplně je určen stylem štětce.

MeasureString – tato metoda měří výšku políčka, které by bylo vykreslené metodou ***DrawString*** pomocí zadaného fontu a šířky políčka. Díky ní zjistíme, jaký prostor zabírá vykreslený text, což umožňuje zjistit plochu rámečku, který text obklopuje.

FillRectangle – tato metoda pro výplň obdélníku se dá využít mnoha způsoby; jeden z nich je vyplnění obsahu pomocí barevného přechodu po zadání počátečního a koncového bodu a barev v těchto bodech. Informace o barevném přechodu jsou uloženy ve štětci, který se metodě předává na vstup. Tuto metodu využíváme pro vykreslení pozadí popisků i celého komiksu.



(Obrázek 12) vykreslení barevného přechodu pomocí dvou zadaných bodů a barev

DrawImage – slouží k vykreslení obrázku ve formě bitmapy. Metodě se zadávají především výsledné souřadnice. ***GDI+*** vykresluje bitmapy tak, aby zachovaly své ***DPI*** rozlišení, což není vždy výhodou, protože většina souborů v sobě nese informaci o ***DPI*** chybnou. Je proto potřeba pro námi používané bitmapy nastavit vždy výslednou šířku a výšku, což tomuto přepočtu zamezí[1].

Pro rychlou práci s bitmapami při aplikaci filtrů bylo potřeba implementovat naše vlastní metody pro přístup k obrazovým datům v nechráněném bloku programového kódu; metody ***SetPixel*** a ***PutPixel*** jsou totiž v praxi velmi pomalé a byly tedy pro naše účely nepoužitelné.

Fonty, které v aplikaci využíváme, jsou získány z internetových stránek[2], které nabízejí mnohé, nejen komiksové, fonty pro použití zdarma. Nachází se v adresáři ***fonts*** ve formátu ***TrueType*** souborů. Program při spuštění načítá všechny dostupné soubory s fonty v tomto adresáři a přidává je do seznamu k již nainstalovaným systémovým fontů.

[1] Microsoft: *Avoiding Automatic Scaling*. http://msdn2.microsoft.com/en-us/library/1bttkazd.aspx [březen, 2007]

[2] http://www.blambot.com, http://www.1000fonts.com

4.1.2 Třída Webclient

Jelikož naše aplikace využívá internet, bylo potřeba využít některou z možností, jak data z internetu zpřístupnit. Tato třída je koncipována jako nástroj s vysokou úrovní abstrakce. Po založení instance této třídy lze rovnou pomocí parametru internetové adresy volat metodu ***DownloadString***, která uloží výsledek ***HTTP*** požadavku do textového řetězce.

Pro univerzálnost programu v různých jazykových prostředí je potřeba vstupní řetězec převést do korektního textového řetězce internetové adresy se zachováním všech textových znaků. Zároveň této třídě nastavujeme jazykové prostředí, použité při analyzování řetězce pomocí výše zmíněné metody ***DownloadString*** tak, že odpovídá jazykovému prostředí operačního systému.

4.1.3 Třída Regex

Pro časté a komplexnější vyhledávání a nahrazování v textovém řetězci již nevystačíme se standardními metodami, jako jsou metody ***IndexOf***, ***Substring***, ***Replace*** apod. Pro tyto účely využíváme podporu regulárních výrazů v našem programovacím prostředí. Konkrétně především ke hledání textových pasáží v ***HTML*** stránkách a extrahování námi požadovaných textových řetězců.

4.2 PHP

Pro dodatečné internetové klientské rozhraní, které uživateli umožňuje využít funkčnost této aplikace bez nutnosti spouštět ji na svém počítači, jsme se rozhodli pro tradiční serverové řešení pomocí programovacího jazyka ***PHP***. Ten nám umožňuje spustit jakoukoliv aplikaci na serveru a výsledek její činnosti zobrazit v ***HTML*** stránce.

Náš skript ze stránky zjistí některé uživatelem nastavitelné hodnoty pro požadovaný komiks, uloží je do souboru ve formátu komiksového projektu a spustí naši aplikaci na serveru. Ta se spustí v serverovém režimu, ve kterém načte komiksový projekt, vygeneruje komiks a uloží na disk výsledný obrazový soubor, který ***PHP*** skript zpět načte a zobrazí v nové ***HTML*** stránce.

Námi vytvořené stránky pouze demonstrují funkčnost a správné provázání webového klienta s naší aplikací, umístěnou na serveru. Je možné vytvořit libovolnou stránku za pomocí kteréhokoliv skriptovacího jazyka, který umožňuje uložení uživatelem nastavitelných hodnot do souboru a spuštění aplikace na serveru.

4.3 Existující filtry a jejich kombinace

Pro napodobení stylu obrázků jsme vybrali některé vhodné obrazové filtry a postupy, které produkují ideální obrázky pro naši aplikaci. Představují špičku mezi komerčně dostupnými grafickými filtry, proto se je budeme ve fázi implementace našeho programu snažit pouze napodobit a ve fázi implementace zvolíme pouze některé z nich. Postupy byly navrhovány v aplikaci ***Adobe Photoshop***, budeme tedy k vysvětlení následujících postupů používat některé termíny a názvy filtrů tohoto programu. U všech postupů předpokládáme nejprve automatickou změnu jasových úrovní u vstupního obrázku. Velikost obrázků byla cca jeden milion obrazových bodů (některé parametry filtrů jsou závislé na obrazovém rozlišení).

Dosažení starého stylu pomocí technik polotónování a posterizace

Technika digitálního polotónování se nabízí jako základ pro napodobení vzhledu starého komiksu. Předem je potřeba fotku převést do podoby obrázku. Zde se osvědčil filtr ***Poster edges***, který redukuje počet použitých barev. Bitmapovou vrstvu s černobílým polotónovým vzorem umístíme nad již posterizovanou vrstvu v režimu ***Hard light*** a doplníme obrysem pomocí vrstvy obsahující filtr ***Glowing edges***, který nachází výrazné hrany v obraze.

Postup v programu *Adobe Photoshop*

- 1. vrstva = (vstup ⇨ filtr: ***Poster edges***, parametry {2, 1, 0})
- 2. vrstva = (vstup ⇨ filtr: ***Halftone pattern***, parametry {2, 4, ***Dot***})
- 3. vrstva = (vstup ⇨ filtr: ***Glowing edges***, parametry {1, 12, 13}) + vyšší kontrast
- výsledek = 3. vrstva (režim ***Multiply***) + 2. vrstva (režim ***Hard Light***) + 1. vrstva

(Obrázek 13) posterizace s polotónováním a hranami

Dosažení noir stylu pomocí technik prahování a překryvem zvolené barvy

Technik prahování je celá řada, nám však nejlépe posloužil filtr ***Torn edges***, který dbá na zachování hran. Po jeho aplikaci je výsledkem černobílý obrázek. Dodatečnou barvu k němu můžeme přidat další vrstvou v režimu ***Linear light***; díky němu se barva objeví i na bílých místech. Tato vrstva obsahuje barvy pouze tam, kde byl ve zdrojovém obraze nalezen odstín požadované barvy.

Postup v programu *Adobe Photoshop*

- 1. vrstva = (vstup ⇨ filtr: ***Torn edges***, parametry {20, 15, 15})
- 2. vrstva = (vstup ⇨ výběr zvolené barvy, inverze výběru, smazání výběru)
- výsledek = 2. vrstva (režim ***Linear light***) + 1. vrstva

(Obrázek 14) prahování s barevným překryvem

Dosažení stylu malby

Dá se jej dosáhnout množstvím filtrů, které napodobují malířské techniky. Tyto filtry našly uplatnění i ve filmové praxi, kde jsou schopny z klasického filmu vyrobit věrohodný kreslený film (vhodným příkladem je film ***Waking life***[1]). Jedna ze tříd těchto filtrů funguje na principu posterizace se zjednodušením nalezených linií. Jejím zástupcem v programu ***Adobe Photoshop*** je filtr ***Cutout***. Pro imitaci tahů štětce použijeme filtr ***Dry brush***. Nakonec vše doplníme vrstvou obrysu podobně jako v případě starého stylu kresby.

Postup v programu *Adobe Photoshop*

- 1. vrstva = (vstup ⇨ filtr: ***Cutout***, parametry {5, 4, 2})
- 2. vrstva = (vstup ⇨ filtr: ***Dry brush***, parametry {2, 9, 1})
- 3. vrstva = (vstup ⇨ filtr: ***Glowing edges***, parametry {1, 12, 13}) + vyšší kontrast
- výsledek = 3. vrstva (režim ***Linear Burn***) + 2. vrstva (režim ***Lighten***) + 1. vrstva

[1] http://www.csfd.cz/film/13227-waking-life/

(Obrázek 15) posterizace s malbou doplněna o hrany

Dosažení stylu upravené fotografie

Pro částečnou stylizaci stačí zvolit vysoký kontrast a potlačení sytosti na nižší úroveň. Další možností je potlačení barevné informace u odstínů, které nejsou shodné s vybraným odstínem.

Postup v programu *Adobe Photoshop*

- 1. vrstva = vstup
- 2. vrstva = (vstup ⇨ ***Desaturate***)
- mezivýsledek = 2. vrstva (režim ***Overlay***) + 1. vrstva
- výsledek = (mezivýsledek ⇨ zvýšení kontrastu, zvýšení ***Gamma***)

(Obrázek 16) režim ***Overlay*** a zvýšení kontrastu

5 Implementace programu

5.1 Běh programu

Ještě než popíšeme postup při implementaci programu, rádi bychom nejprve stručně popsali, jak program funguje. Program nabízí dva režimy běhu; uživatelský a serverový. Aplikace z příkazové řádky zjistí, ve kterém z těchto módu má být spuštěna. Oba režimy běhu mají společné následující první kroky.

- Načtení základní konfigurace ze souboru
- Načtení dostupných ***TrueType*** fontů
- Nastavení uživatelem nastavitelných hodnot na standardní hodnoty

5.1.1 Uživatelský režim

V uživatelském režimu se po načtení základních nastavení spustí okno aplikace, neboli grafické uživatelské rozhraní. To se dále stará o veškerou komunikaci s renderovacím systémem (popsaným dále), zadává mu požadavky a získává od něj mimo jiné především vytvořený komiks.

Typicky probíhá běh v tomto režimu následujícím způsobem. Uživatel vyplní požadovaný název komiksu, své jméno (nebo jiné informace týkající se identity tvůrce komiksu) a především komiksový text. Dále si zvolí obrazové rozlišení komiksu a rozvržení komiksové stránky. Poté tlačítkem pro generování nové posloupnosti náhodných hodnot generuje takové hodnoty, při kterých je rozvržení komiksových políček a rozdělení textu do komiksových popisků subjektivně nejlepší. Automatické rozdělování textu lze uživatelem ovlivnit pomocí textových značek, principem převzatých z jazyka ***HTML*** („*** ***“ označuje místo, kde se nemá text dělit a „***&hbsp***“ naopak místo, kde se má text povinně rozdělit). Změny rozložení jsou sice umožněny i po vložení obrázků, ale především jen pro jemné doladění některých elementů. Obrázky by se totiž musely po každé změně rozložení získat z internetu znovu a převzorkovat, což by trvalo dlouho.

Když je uživatel s tímto rozvržením spokojen, stiskne tlačítko ***Generate***. Tím se spustí v jiném programovém vlákně proces, který pro komiksové pozadí a každé komiksové políčko obstará dvě úlohy. První úlohou je nalezení (maximálně) tří nejvhodnějších klíčových slov, která charakterizují text daného komiksového políčka; pokud analýza vyhodnotí všechna slova jako nevhodná, je políčku přiřazeno klíčové slovo nalezené v nadpisu komiksu. Druhou úlohou je nalezení náhodného a graficky dostatečně kvalitního obrázku ve zvolené online vyhledávací službě pomocí nalezených klíčových slov. Během tohoto procesu může uživatel měnit pouze hodnoty, které jsou v této fázi dostupné; mezi nimi jsou především nastavení stylu, barev, fontů, apod.

Po skončení tohoto procesu se v náhledovém okně nachází komiks i s nalezenými obrázky. Proces lze přerušit stisknutím tlačítka ***Cancel***; v tom případě jsou ponechány dosud nalezené obrázky a program se vrátí se do stavu běžné činnosti.

V další fázi je často potřeba nahradit nevhodně nalezené obrázky jinými. K tomu slouží v panelu nastavení záložka výsledků hledání (***Results***), ve které lze pro pozadí a každé políčko nalézt nový obrázek dle zadaných klíčových slov. Lze také zadat klíčová slova manuálně, případně vložit konkrétní internetovou adresu nového obrázku, nebo přímo obrázek ze souboru na disku.

Poslední nastavení vzhledu se týká grafických filtrů aplikovaných na obrázky. Nezávisle na sobě jsou aplikovány filtry obarvení komiksu (především potlačení sytosti barev s výjimkou uživatelem definovaného odstínu vybrané barvy) a překrytí obrázku polotónovým vzorem, který se z obrázku získá pomocí binárního prahování.

Nakonec nezbývá, než výsledný komiks uložit ve formě obrazového souboru ve formátu ***Jpeg***, ***Bmp***, či ***Png***. Pokud chce uživatel ve tvorbě komiksu pokračovat jindy, uloží si projektový soubor a po dalším spuštění aplikace jej načte. Obrázky se při ukládání projektu uloží na disk, aby byla později při načtení zajištěna jejich dostupnost (pokud by byly z internetu odstraněny).

Následuje shrnutí dostupných operací v tomto režimu, které bylo třeba implementovat:

- vytvoření nového projektu, uložení stávajícího projektu, načtení uloženého projektu
- uložení komiksu do grafického souborového formátu
- přepnutí mezi rychlým, méně kvalitním interaktivním módem a módem s vysokou kvalitou obrazu ve výsledné kvalitě renderování
- nastavení uživatelem nastavitelných hodnot pomocí ovládacích prvků v panelu nastavení
- generování nové posloupnosti náhodných hodnot použitých při generování rozmístění komiksových políček a rozdělení vět do popisků
- náhodné rozmístění uživatelem nastavitelných hodnot pro různé sekce v panelu nastavení
- nalezení náhodného článku (ze stránek typu ***Wikipedia***[1], uvedených v konfiguraci) a extrahování jeho nadpisu a textu
- spuštění na pozadí běžícího procesu, jenž automatizuje celý proces generování komiksu tím, že ze zadaného nadpisu, textu, rozvržení komiksu a rozdělených vět najde analýzou nejvhodnější klíčová slova pro vyhledání a najde z nich pro každé komiksové políčko obrázek
- změna libovolného nalezeného obrázku a jeho náhrada za lokální grafický soubor
- interaktivní změna pozice a velikosti komiksových elementů (především obrázků, nadpisu a titulků) přímo v náhledovém okně komiksu

[1] http://en.wikipedia.org/wiki/Main_Page

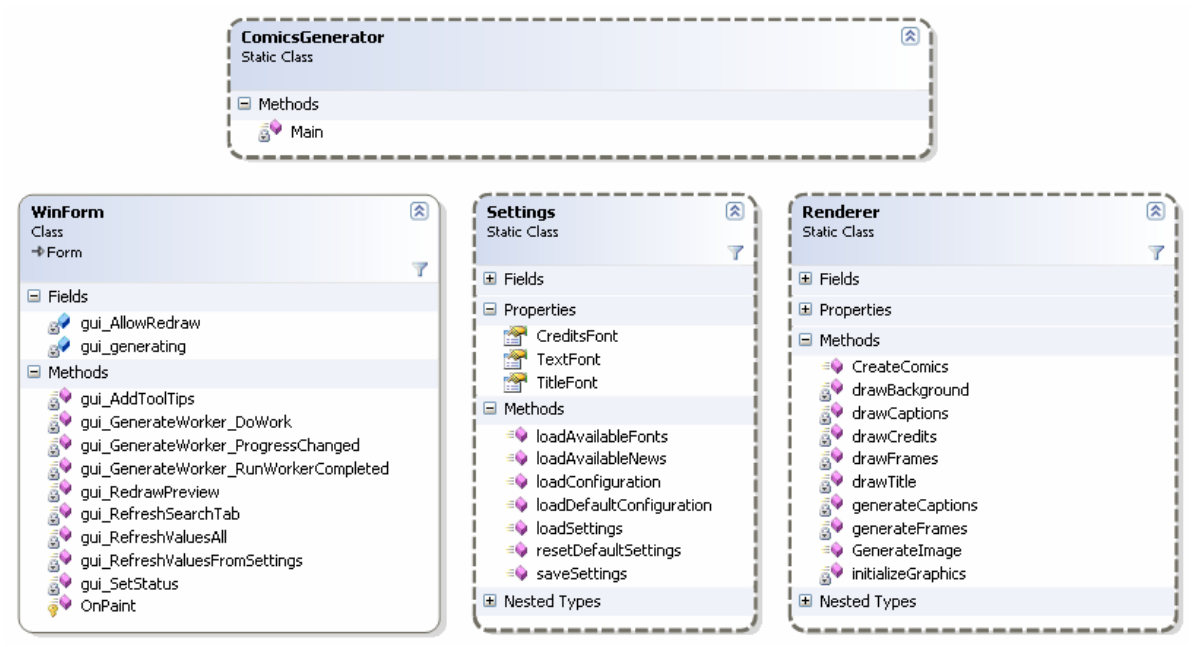
5.1.2 Serverový režim

Program v serverovém režimu načte projektový soubor dodaný správcem serveru, který slouží jako základní šablona v případě, že webový klient neumožňuje nastavit všechny dostupné hodnoty. Hodnoty, které umožňuje nastavit, jsou dodané v druhém projektovém souboru, který skriptovací jazyk vytvoří. Díky tomu, že načtení projektových souborů přepisuje vždy jen hodnoty, které jsou v projektovém souboru uloženy, se dá nastavení hodnot vrstvit.

Možnosti nastavení hodnot jsou závislé na tom, jak detailně je vypracováno internetové klientské prostředí. Teoreticky umožňuje stejné možnosti jako uživatelský režim, jde jen o to, jak pestré jsou možnosti tvorby projektového souboru, který obsahuje veškeré nastavitelné hodnoty potřebné pro vytvoření komiksu. Po načtení všech nastavitelných hodnot se vygeneruje komiks i s obrázky. Po vytvoření je komiks uložen do souboru ve formátu ***Jpeg***.

5.2 Diagram tříd

Pro lepší představu provázanosti námi implementovaných tříd uvádíme jejich diagram. Hlavní třída aplikace je třída ***ComicsGenerator***. Ta slouží pouze jako vstupní bod programu a dále dle výše zmíněných dvou režimů spustí buď okno aplikace (***WinForm***), nebo pouze zavolá dále popsané základní metody třídy ***Renderer***, uloží obrázek a ukončí se. Statická třída ***Settings*** slouží jako společné úložiště uživatelem nastavitelných hodnot a obstarává načtení uživatelských konfigurací, dostupných fontů a manipulace s projektovým souborem.



(Obrázek 17) diagram tříd spolu s jejich nejzajímavějšími metodami

Dále stručně uvedeme tři nejčastěji využívané pomocné třídy. Třída ***Frame*** v naší aplikaci reprezentuje komiksové políčko. Obsahuje jeho souřadnice, rozměry, posun od automatického umístění (instance třídy ***Offset***), adresu obrázku, klíčová slova pro toto políčko, uložené bitmapy obrázku pozadí a maximálně dvě instance třídy ***Caption*** (popisek). Ta v sobě obsahuje text, své souřadnice a rozměry a určení jednoho ze dvou typů popisku.

(Obrázek 18) diagram pomocných tříd: ***Frame*** (políčko), ***Caption*** (popisek), ***Offset*** (odchylka pozice)

5.3 Renderovací systém

Základní komponentou, která se stará o vytvoření celého komiksu, je třída ***Renderer***, která představuje náš renderovací systém. Z implementačního hlediska se jedná o statickou třídu, jejíž metody jsou volány buď z hlavního programu v případě serverového režimu, nebo z okna aplikace v druhém případě. Hlavním využitím této třídy je volání její metody ***CreateComix***, která vytvoří komiks na základě uživatelem nastavitelných hodnot, které jsou uloženy ve třídě ***Settings***. Dále implementuje funkce pro vyhledání vhodných klíčových slov reprezentujících text daného políčka a funkce, které z těchto klíčových slov najdou vhodnou internetovou adresu obrázku.

Jelikož je grafické rozhraní ***GDI+*** v mnoha případech opravdu výrazně pomalejší než ***GDI*** (důvodem je nedostatečná podpora hardwarové akcelerace), byli jsme nuceni zavést dva režimy vykreslování. První režim reprezentuje vysokou kvalitu zobrazení, která se však nehodí pro interaktivní manipulaci s komiksem a slouží především pro finální uložení obrázku. Uživatel má však možnost se do tohoto režimu kdykoliv přepnout a pracovat v něm. Druhý režim používá méně kvalitní filtrace obrazu a má mnohem menší rozlišení obrazu (závislé na původním rozlišení, tj. malé obrázky v tomto režimu již kvůli čitelnosti písma nemění své rozlišení, rozměrné obrázky však ano). Také barevné filtry na obrázcích probíhají v tomto režimu podvzorkovaně. Navenek však pro uživatele při přepnutí režimu komiks v náhledovém okně zachovává zobrazovanou pozici i velikost.

(Obrázek 19) náhledové okno v režimu nízké kvality (vlevo) a vysoké kvality (vpravo)

Protože by bylo vykreslování všech komiksových elementů při interaktivních funkcích velmi pomalé, provádí se většina generovacích a vykreslovacích operací jen tehdy, když jsou zapotřebí. Například filtrace obrazu probíhá pouze tehdy, když uživatel nastavuje hodnoty související s touto filtrací. K tomu, aby renderovací systém věděl, které operace má provádět, slouží proměnná, do které okno aplikace nastavuje typ právě prováděné činnosti.

Jelikož má uživatel možnost kdykoliv změnit nastavení rozlišení komiksového obrázku, bylo potřeba zajistit, aby při těchto změnách zůstaly veškeré relativní vzdálenosti a velikosti elementů zachované. Toho je dosaženo tím, že všechny uživatelem nastavitelné hodnoty, týkající se rozměrů, jsou relativní vůči rozlišení a typu rozvržení komiksu (komiksová stránka či strip). Renderovací systém při těchto změnách však přepočítá skutečné vzdálenosti a uchová je pro další použití, aby nemusely být počítány v každém cyklu překreslení obrazu znovu.

5.3.1 Hlavní vykreslovací metoda

CreateComix se nazývá hlavní vykreslovací metoda třídy ***Renderer***, která je volána pokaždé, kdy je uživatelem změněna některá z hodnot nastavení. Jsou v ní volány další pomocné metody, potřebné k vytvoření komiksu. Ty se dají rozdělit na metody, které se starají o vygenerování různých souřadnicových hodnot komiksových elementů, klíčových slov a obrázků a na metody, které jednotlivé elementy vykreslí.

Nejprve se zkontroluje, zda nebyla změněna některá z hodnot, která by způsobila změnu rozlišení obrazu. Pokud ano, provede se nová inicializace grafických objektů a bitmap pomocí metody ***InitializeGraphics***. Dále se vygeneruje posloupnost pseudonáhodných čísel, díky kterým má komiks jedinečný vzhled i při jinak stejných parametrech nastavení. Tato posloupnost je vygenerována pomocí semínka, které je stále stejné, dokud uživatel nestiskne tlačítko pro vygenerování nového semínka. Toto semínko je uloženo v projektovém souboru, aby po jeho načtení byl komiks totožný s uloženým komiksem.

5.3.2 Generující metody

Nyní se zaměříme na metody, které obstarávají generování pozic a velikostí jednotlivých komiksových elementů. Ty jsou poté k dispozici metodám, které tyto elementy vykreslují. Často se zde vyskytuje použití náhodných čísel, generovaných výše uvedenou posloupností náhodných čísel.

Generování políček

Metoda ***GenerateFrames*** generuje velikosti a umístění komiksových políček dle uživatelem nastavitelných hodnot, které souvisí s grafickým rozvržením komiksu. Vypočtené hodnoty souřadnic se ukládají do pole instancí výše uvedené třídy ***Frame***. Toto pole reprezentuje všechna políčka aktuálního komiksu. Při novém generování se pouze přepisují staré hodnoty v těchto instancích a ponechávají se v nich uložené internetové adresy a bitmapy. Důvodem je ponechání možnosti, aby uživatel mohl libovolně měnit rozvržení stránky (a tím i počet políček) bez toho, aby byly smazány obrázky z políček, které aktuální počet políček převyšují.

Základní rozložení políček vychází z konceptu mřížky, která má nastavitelné rozměry počtu políček v horizontálním a vertikálním směru. Komiksová políčka jsou poté umístěna do této mřížky. Od okrajů stránky komiksu je mřížka vzdálena dle uživatelem nastavitelné hodnoty odsazení mřížky. Pokud by každé políčko zabralo právě jednu buňku mřížky, byl by komiks jednotvárný. Proto jsme zpočátku implementovali algoritmus, který každý řádek této mřížky náhodně rozdělil na nestejně veliká políčka. Ta poté většinou zabírala pozici jedné až tří buněk mřížky v horizontálním směru. Při pokusu o univerzálnější velikost mřížky a přidání možnosti o netradiční velikost políček ale přesto vypadalo toto rozložení jednotvárně, jelikož měl každý řádek stále stejnou výšku. Navíc jsme chtěli uživateli umožnit přesně nastavit míru, do jaké mají mít políčka různorodou velikost. Bylo tedy potřeba změnit algoritmus od základu.

Po několika pokusech o generování takového rozmístění pomocí jednoduchých pravidel jsme zavedli rozhodování dle tabulky pravděpodobností (Tabulka 1). V ní je uvedeno, jakou mají jednotlivé rozměry políčka (v jednotkách buněk mřížky) pravděpodobnost výskytu při dané uživatelem nastavitelné hodnotě s ohledem na některá pravidla a speciální případy (výjimky u okrajů a některá další pravidla, která souvisí s tím, aby text správně pokračoval zleva doprava a shora dolů). Tato uživatelem nastavitelná hodnota může nabývat hodnot 0 % (všechna políčka budou mít velikost 1×1 jednotek mřížky) až 100 % (políčka budou mít větší a různorodější velikosti), viz (Obrázek 20). V tabulce jsou uvedené hodnoty pro 0 %, 50 % a 100 %. Hodnoty mezi nimi se lineárně interpolují. Tabulku lze upravit v konfiguračním souboru a zadat jiné velikosti políček a jejich pravděpodobnosti.

(Obrázek 20) velikosti políček při hodnotě 0 % (vlevo) a 100 % (vpravo)

Šířka	Výška	0 %	50 %	100 %
1	1	100	60	0
2	1	0	18	2
3	1	0	10	3
1	2	0	6	7
4	1	0	3	13
5	1	0	1	15
2	2	0	0	10
6	1	0	0	15
3	2	0	0	15
3	3	0	0	10

(Tabulka 1) tabulka velikosti políček

Některé buňky je možné vynechat a vytvořit tím prázdné místo; jejich procento je přímo úměrné procentu zadaném uživatelem. V extrémním případě (hodnota 0 %) není vygenerována žádná buňka. Dále se upraví konkrétní pozice a rozměry políčka dle uživatelem nastavitelné hodnoty šířky žlábku (v horizontálním i vertikálním směru) mezi políčky. K pozicím se přičtou hodnoty, náhodně vygenerované pomocí uživatelem nastavitelných hodnot variability umístění políček a šířky žlábku.

Rozdělení komiksového textu

Nyní je potřeba do políček vhodně rozmístit text komiksu. V ranné fázi implementace (v semestrální práci) jsme použili metodu, kdy se náhodně políčkům přiřazovaly věty (respektive se jim přiřazoval počet vět), dokud nebyl maximální počet vět vyčerpán. Poté se v každém políčku nahradil symbol, který označoval počet přiřazených vět, za skutečné věty. Věty byly tvořeny řetězci o určité minimální velikosti, zakončené tečkou nebo čárkou; to bylo omezení, které jsme chtěli odstranit. Pokud byla vstupním textem například báseň nebo jiný útvar, který nebyl rozčleněn do tradičních vět, rozdělení neproběhlo dle uživatelova očekávání.

Algoritmus nyní probíhá na principu rekurzivního členění textu na dvě poloviny dle několika kritérií. Oddělovačem není již jen tečka či čárka, ale množina znaků, které mají určenou svoji prioritu pro dělení textu. Tato množina, implementovaná pomocí pole, lze uživatelem nastavit v konfiguračním souboru a umožňuje dělení textu i pro jazyky, které používají zcela jiná interpunkční znaménka. Standardně je oddělovačem znak nového řádku, tečka, vykřičník, otazník, čárka a nakonec mezera. Ta umožní text rozdělit v případě, že je věta příliš dlouhá a není k dispozici žádný jiný oddělovací znak.

Rozdělení pomocí mezer bohužel nefunguje v případě jazyků, které mezi slovy žádnou mezeru nepoužívají (některé asijské jazyky jako čínština, japonština, korejština, atd.). Algoritmus rozdělení je již v tomto případě složitý a jeho implementační řešení je popsáno například v článku publikovaném na internetových stránkách ***CodeProject***[1]. V konfiguračním souboru je možné určit, od kterého oddělovacího znaku je již potřeba text rozdělit pomocí dodatečného textového řetězce „**…**“. Standardně je toto chování nastavené pro znak čárky a mezery.

Na začátku metody pro rozdělení textu je nejprve pomocí rozměrů písmene „M”, vykresleného aktuální velikostí fontu, zjištěna průměrná maximální délka textu v jednom popisku komiksového políčka. Pokud je vstupní komiksový text delší než práh, který zhruba odpovídá čtvrtinovému naplnění políčka, nebo pokud text obsahoval příliš mnoho textových značek pro povinné rozdělení textu a text se tedy nemůže vejít do komiksu celý, končí komiks slovy „***to be continued…***”. Tento textový řetězec lze nahradit v konfiguračním souboru za jiný.

Výše zmíněné rekurzivní funkci se předává vstupní textový řetězec (upravený dle jednoduchých formátovacích pravidel), požadovaný počet rozčleněných částí a index na pole s komiksovými políčky. Index se používá proto, aby bylo možné upravit délku rozděleného textu dle velikosti políčka, kterému náleží. Funkce vrací pole již rozdělených textových řetězců.

Na začátku funkce se zkontroluje, zda počet požadovaných částí není roven jedné. Tím nastane ukončení rekurze, vrátí se tedy celý řetězec. Výjimkou je, pokud řetězec obsahuje speciální textovou značku pro povinné rozdělení; v tom případě se text rozdělí v místě výskytu této značky. Pokud je počet požadovaného dělení vyšší než jedna, zjišťuje se přítomnost oddělovacího znaku od nejvyšší priority k nejnižší. Pokud je oddělovací znak nalezen a je jich více, pak se rozhodne o konkrétním oddělovacím znaku tak, aby do rekurzivního volání pro levou i pravou stranu textového řetězce vstoupilo množství textu přibližně úměrné velikostem políček, ke kterým náleží. Pokud je například zjištěno, že na vstupní text připadají čtyři políčka, z nichž první dvě mají velikost 2×2 jednotek mřížky a druhé dvě pouze velikost 1×1, pak se text rozdělí pokud možno tak, že 4/6 textu vstupuje do rekurze levé části textu a zbylé 2/6 do pravé části.

Poté, co se získá pole textových řetězců pro popisky, je potřeba nahradit některé pomocné textové značky. Jedna z nich je textový řetězec, který je k textu připsán, pokud byl text rozdělen pomocí oddělovačů, které mají stejnou prioritu jako čárka a nižší. V tom případě se text doplní o textový řetězec „**…**”. Dále se nahradí speciální značka pro nedělitelný text za znak mezery.

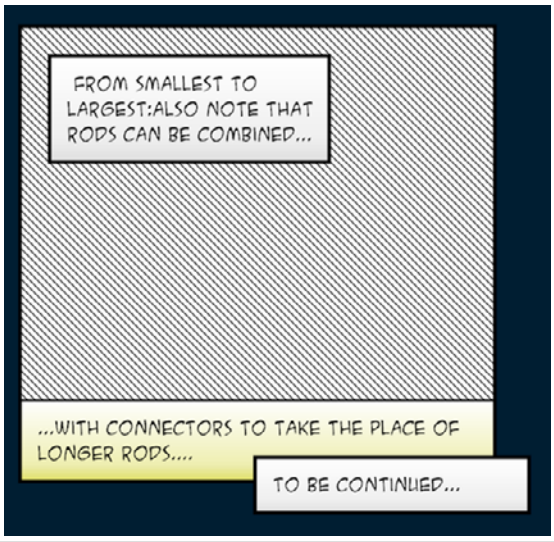
Výsledné popisky se poté přiřadí políčkům. Nejprve se přiřazuje tak, aby každé políčko mělo nejvýše jeden popisek. Pokud zbude popisků více, přiřazují se následně k políčkům i druhé popisky na základě náhodně zvoleného políčka.

[1] Cardinal J.: *Unicode compliant multilingual word breaker*. http://www.codeproject.com/csharp/breaker.asp [duben, 2007]

Generování popisků

V další fázi se generují pozice a rozměry popisků. Program generuje dva typy popisků; v naší implementaci je nazýváme běžné a alternativní. Běžné popisky jsou umístěny volně na ploše políček s různorodou velikostí, zatímco alternativní popisky jsou pevně přichyceny k vnitřnímu okraji políčka. Oba typy mají své vlastní nastavení vizuálního stylu. Při generování popisků je potřeba mezi nimi rozlišovat. V každém políčku, které obsahuje dva popisky, je potřeba je vhodně rozmístit tak, aby se nepřekrývaly a aby zachovávaly pořadí vět v textu. K tomu je vypracována sada jednoduchých pravidel, která určují, kde se může popisek v políčku vyskytovat. Děje se tak v rámci pomyslné mřížky o rozměrech 2×2, která políčko reprezentuje.

Uživatel má možnost definovat ve škále od nuly do sta procent, zda chce zarovnávat popisky tradičně vlevo nahoře nebo naopak vpravo dole. Mezi těmito dvěma polohami se políčka náhodně rozmísťují s danou váhou. Po tomto hrubém rozmístění se počítají konkrétní souřadnice a velikosti políček pomocí metody ***MeasureCaptionSize***, která vytvoří pravoúhlý podklad popisku tak, aby přesně obepínal text (s nastavitelným odsazením od textu) a aby poměr stran nenabýval extrémních a vizuálně nepoužitelných hodnot. Pozice popisku se v případě běžného typu popisku určí podle uživatelem nastavitelné hodnoty vzdálenosti popisku od okraje políčka a náhodného rozptylu této hodnoty.



(Obrázek 21) shora dolů – běžný popisek, alternativní popisek, popisek při překročení maximální délky textu

Pozice nadpisu a titulků

Při změnách rozlišení nebo stisku tlačítka pro automatickou pozici je potřeba vypočítat automatické velikosti a pozice nadpisu a titulků. Nejprve se určí rozměry těchto elementů dle zadaného fontu a podle nich se rozmístí dle zvoleného odsazení mřížky rozvržení. Pokud text titulků obsahuje pouze jeden řádek, algoritmus se jej pokusí vtěsnat mezi nadpis a mřížku; v opačném případě je umístěn vpravo od nadpisu, zarovnaný na pravý okraj mřížky.

5.3.3 Metody pro vyhledání obrázku

Nalezení klíčových slov

Pro vyhledání vhodného obrázku na pozadí políčka je nejprve potřeba analyzovat text, který je v políčku obsažen, a najít v něm nejvýše tři klíčová slova, která mají největší informační hodnotu a pokud možno text nejlépe charakterizují. Náš algoritmus pracuje na principu bodového ohodnocení všech slov v textu políčka. Pokud políčko neobsahuje žádný text, přiřadí se mu klíčová slova nalezená z nadpisu komiksu, stejně jako je tomu v případě obrázku na pozadí.

Dále se v komiksovém textu naleznou syntakticky správně zapsaná slova, charakterizovaná regulárním výrazem „*\w+*“. Dále se zjistí, zda již stejné slovo v políčku nebylo ohodnoceno; pokud ano, již se dále nepočítá. V dalším kroku se zjistí, zda existuje pro dané slovo alespoň určitý minimální počet nalezených obrázků na internetu. To je první možnost jak zjistit, zda je slovo alespoň trochu smysluplné a zároveň jsou tím odstraněna slova, která by stejně i při následném vysokém bodovém ohodnocení neprodukovala žádný nalezený obrázek ke stažení. Pokud slovo splní výše zmíněná kritéria, začne se pro něj počítat bodové ohodnocení.

Nejprve se slovo ohodnotí dle počtu nalezených obrázků, regulovaného lineární křivkou, která zajistí, aby se rozlišovalo především rozmezí mezi prvními stovkami až desetitisíci nalezenými výsledky. Důvodem pro to je fakt, že velmi často používaná slova jako jsou spojky a některé další slovní druhy, které mají až stovky miliónů odkazů a které chceme vzápětí odfiltrovat, by byly ohodnoceny příliš vysokým počtem bodů. Přitom u slov, která označují název nebo se používají pouze v užším okruhu lidí (například jsou psána v málo užívaném jazyce) je pro nás i počet okolo několika stovek dostatečnou známkou smysluplnosti daného slova.

V další fázi se snažíme zjistit, zda slovo náleží ke slovnímu druhu, který považujeme za slovní druh s vyšší informační hodnotou. Například běžně mají podstatná, přídavná jména a slovesa vyšší informační hodnotu než ostatní slovní druhy. Slovní druh lze zjistit na internetu pomocí vhodného slovníku či tezauru. Zatím tyto služby bohužel nejsou příliš kvalitní a spolehlivé. My jsme zvolili pravděpodobně nejznámější anglický online slovník[1] (a dodatečně pro porovnání i anglický tezaurus[2]), kterému pomocí ***HTTP*** metody ***GET*** pošleme požadované slovo a z výsledné odpovědi ve formě ***HTML*** stránky zjistíme, o který slovní druh se jedná.

Abychom umožnili použití slovníku i pro ostatní jazyky, je možné vytvořit nový konfigurační soubor s jiným online slovníkem spolu s textovými řetězci, které označují začátek a konec textového bloku ***HTML*** kódu, ve kterém se nachází požadovaný název slovního druhu. Dále se zadávají dva textové řetězce, které označují slovní druh s nejvyšší a poloviční prioritou (standardně podstatné a přídavné jméno). Mezi dostupnými slovníky lze v okně aplikace kdykoliv přepínat.

[1] http://dictionary.reference.com

[2] http://thesaurus.reference.com

V této fázi je primárním účelem snížit již dosažené bodové ohodnocení slovům, představující slovní druh, který nás nezajímá (částice, spojky, příslovečná určení, atd.). Pokud není slovo ve slovníku nalezeno, nic to pro jeho bodové hodnocení neznamená. V opačném případě, kdy je slovo nalezeno a nejedná se o slovní druh s nejvyšší prioritou, je bodové hodnocení děleno koeficientem 2 u slovního druhu s poloviční prioritou a koeficientem 5 v ostatních případech. Hledání ve slovníku lze vypnout (standardně je vypnuto), což výrazně urychlí vyhledávání klíčových slov a navíc v mnoha jazycích nemá použití anglického (standardního) slovníku význam.

Nyní by již měly mít vyšší hodnocení jen známá podstatná či přídavná jména a slova, ke kterým tezaurus nenašel slovní druh, a jedná se tedy nejspíše o název či jméno. Zbývá nepatrně zvýšit hodnocení podle délky textového řetězce, protože se očekává, že v mnoha jazycích má delší slovo větší informační hodnotu.

Hodnocení slova, které bylo získáno pomocí slovníku, se pro příští rychlejší zpracování uloží do souboru ***keywordsScoreCache.dat***, ve kterém se uchovávají bodová hodnocení již dříve nalezených slov. Každé slovo v nově generovaném komiksu se nejprve vyhledá v tomto souboru, a pokud je nalezeno, přečte se z něj bodové hodnocení a dále se již nepočítá.

Nalezení obrázků

Když jsou nyní k dispozici klíčová slova pro každé políčko, je možné z nich začít vyhledávat náhodné obrázky pomocí online vyhledávací služby. Ta se dá definovat pomocí nového konfiguračního souboru. Mezi těmito konfiguračními soubory lze v okně aplikace kdykoliv přepínat.

Standardně se využívá vyhledávací služba ***Google images***, která poskytuje největší indexovanou databázi obrázků na internetu. Při její implementaci jsme se inspirovali projektem z ***CodeProject***[1]. Druhá, námi zahrnutá vyhledávací služba je ***Devianart***[2], která je sice výrazně pomalejší, zato však nabízí vhodnější obrázky do kresleného komiksu.

Nejprve se zkusí obrázek vyhledat pomocí všech nalezených klíčových slov zároveň (včetně uživatelem zadaných dodatečných klíčových slov). V případě nulového počtu nalezených obrázků se postup opakuje s menším počtem klíčových slov, dokud není nalezen alespoň jeden obrázek.

Vyhledávání obrázků pracuje podobně jako u slovníku. Pomocí metody ***GET*** se pošle vyhledávací službě požadavek na vyhledání obrázků k příslušnému klíčovému slovu a z výsledné odpovědi se naleznou konkrétní odkazy na obrazové soubory. (Obrázek 22) ukazuje příklad textového řetězce tohoto požadavku.

```
http://images.google.com/images?q=halftone&start=60&filter=1
```

(Obrázek 22) ukázka požadavku na vyhledávací službu ***Google images***

[1] Assayag I.: *An API for Google image search*. http://www.codeproject.com/cs/library/google_image_search_api.asp [únor, 2007]

[2] http://www.deviantart.com

Požadované klíčové slovo je v případě služby ***Google images*** uvedeno textovým řetězcem „***q***=“. Index na konkrétní stránku nalezených výsledků je uveden textovým řetězcem „***start***=”. Dále je možné v konfiguračním souboru doplnit dodatečné textové řetězce (například „***filter=1***“ pro filtrování obsahu nalezených stránek). Ve výsledné ***HTML*** stránce poté náhodně hledáme internetové odkazy na obrázky pomocí konfigurovatelného regulárního výrazu (například ***Google images*** již v průběhu nejméně jednou změnil strukturu výsledné stránky, proto je potřeba mít možnost tento regulární výraz konfigurovat). Pokud je nalezený obrázek příliš malý, zvolí se náhodně další nalezená adresa, dokud není úspěšně obrázek uložen do bitmapy daného políčka.

5.3.4 Vykreslovací metody

Vykreslení pozadí

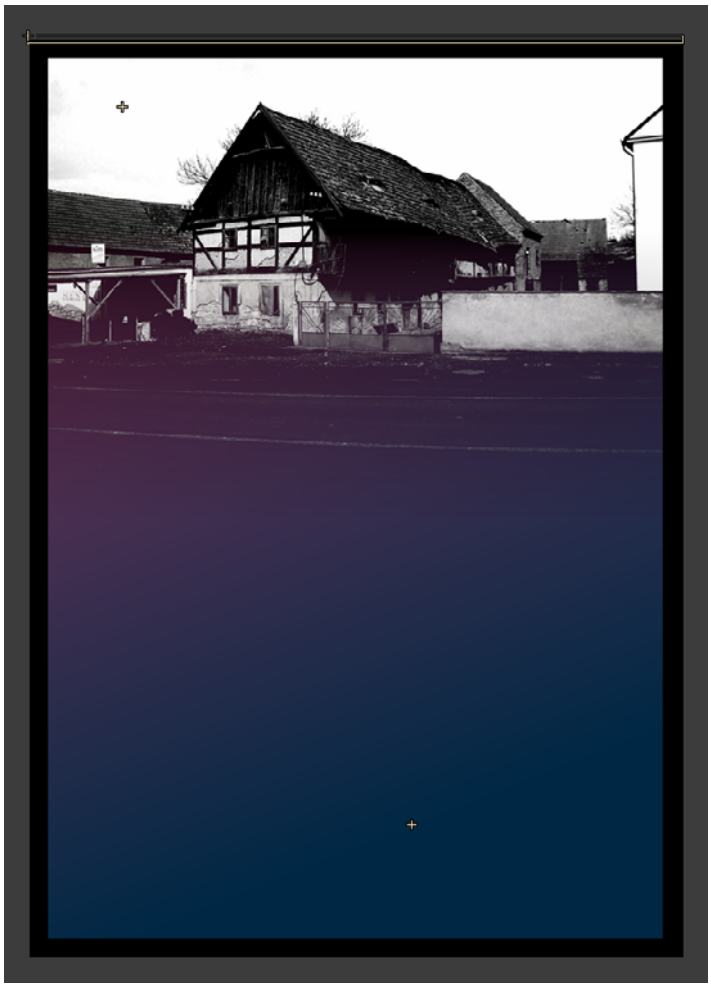
První vykreslovací vrstvou je pozadí. V naší implementaci se skládá ze tří částí; z barevného přechodu, obrázku a orámování. Barevný přechod je implementován jako výplň celého kreslícího plátna komiksu pomocí štětce, který má nastavený režim barevného přechodu. Uživatel si interaktivně zvolí počáteční a koncový bod tohoto přechodu a také jejich barvy.

Zde jsme narazili na technický problém spojený s barevným přechodem v místě za koncovými body. ***GDI+*** nabízí různé režimy opakování přechodu, z nichž bohužel právě ten, který jsme potřebovali a který by ponechal stejné barvy za okrajem přechodu (režim ***Clamp***), nejde pro tuto metodu výplně použít. Proto jsme tento efekt simulovali tím, že jsme od sebe koncové body vzdálili na vzdálenost úhlopříčky celého plátna a skutečné body, ve kterých měl přechod začít a skončit, jsme dopočítali.

Na tento barevný podklad se nanese obrázek pozadí na automatické umístění, které lze poté uživatelem interaktivně měnit. Obrázek na pozadí je konstrukčně implementován stejně jako políčko (typ ***Frame***), proto na něj lze aplikovat shodné postupy při jeho umístění i filtraci jeho bitmap.

Pokud bychom vykreslili obrázek běžným způsobem, působily by okraje v horní a spodní části obrázku rušivě (šířka obrázku se standardně nastaví na šířku komiksu, proto boční okraje rušivě nepůsobí). Z toho důvodu jsme implementovali metodu pro vytvoření masky v alfakanálu bitmapy. Tato metoda (***Filter_bgAlphaBlend***) vytvoří v bitmapě alfakanál pomocí uživatelem nastavitelných hodnot vrchního a spodního umístění hranice transparentního přechodu. Obrázek tedy na vrchním a spodním konci plynule splyne s barevným pozadím.

Na obrázek se mohou dle nastavení uživatele aplikovat buď filtry, které jsou právě nastavené u obrázků políček, veškeré filtry, a nebo se obrázek nanese bez aplikace jakýchkoliv filtrů. Nakonec se vykreslí jednoduché orámování komiksu pomocí metody ***DrawRectangle*** s uživatelem zvolenou barvou a tloušťkou pera.



(Obrázek 23) pozadí komiksu – barevný přechod, obrázek a orámování

Vykreslení komiksových políček

Na hotové pozadí se v následujícím pořadí nanesou obrázky políček, alternativní popisky, okraje alternativních popisů, okraje políček, a nakonec běžné popisky i s jejich okrajem. Toto pořadí je nezbytné pro správné vrstvení komiksových elementů. Na obrázky komiksových políček se aplikují obrazové filtry, které podrobněji popíšeme v příslušné podkapitole. Při přidání nového obrázku do komiksu se každý obrázek zmenší na velikost políčka tak, aby jej s přesahem (kvůli případným artefaktům na kraji obrazu) vyplňoval (metodou ***best-fit***[1]).

Uživatel může obrázek interaktivně přemístit a změnit jeho velikost, pokud mu nevyhovuje automatické zarovnání doprostřed. Při vykreslení obrázku je tedy potřeba vykreslit jen tu část obrázku, která je uvnitř políčka. ***GDI+*** neposkytuje metodu, ve které by bylo možné zadat ořezávací obdélník, je proto třeba vytvořit novou bitmapu, která obsahuje pouze tento výřez a tu dále zobrazovat. Jelikož by bylo třeba při každém vykreslení komiksu vytvořit nový výřez, je v každém políčku tento výřez uložen a přepočítá se pouze při interaktivní manipulaci s daným políčkem.

Vykreslení alternativních popisků je shodné s vykreslením běžných popisků (viz dále). Okraje jsou vykresleny stejným způsobem jako okraje pozadí komiksu.

[1] Powell R. W.: *BobPowell.net*. http://www.bobpowell.net/bestfit.htm [březen, 2007]

Jelikož běžné popisky mohou být umístěny mimo prostor políčka, jsou vykreslovány ze všech elementů políčka až na závěr. Z jejich vertikálního rozměru se určí výška lineárního barevného přechodu, použitého na pozadí popisku. Poté je zvoleným fontem s uživatelem nastaveným odsazením vykreslen text popisku. Běžný typ popisků nabízí pro pestřejší variabilitu vzhledu dvě nastavitelné barvy a tloušťky okrajů (vnitřní a vnější).

Vykreslení titulků a nadpisu

Nakonec se jako poslední vrstva vykreslí titulky a po nich nadpis komiksu. Před vykreslením je potřeba v textu nahradit každý znak „|“ (speciální znak označující nový řádek) skutečným znakem nového řádku. Nejprve se vykreslí obrys textu a poté jeho výplň. Obrys jsme nejprve implementovali metodou, při které se text vykresloval okolo svého středu v určité vzdálenosti, která souvisela s uživatelem nastavitelnou hodnotou tloušťky obrysu. Tato metoda věrně orámovala text, ale přestože jsme použili některé optimalizace (pouze některé úhly v závislosti na daném koeficientu kvality a jim náležící předpočítané hodnoty funkcí sinus a kosinus), nebyla příliš rychlá a neumožňovala další nastavení, jako například typ okrajů.

Nakonec jsme zvolili vykreslování obrysu textu pomocí ***GDI+*** metody ***AddString***, která umožňuje z vektorového textu vytvořit objekt typu ***Path***; ten se dá následně obtáhnout zvoleným perem. Pro správné vykreslení obrysu u některých písmen určitého fontu bylo zapotřebí nastavit pomocí parametru ***MiterLimit*** prahovou hodnotu úhlu, při kterém nedojde k zašpičatění rohu cesty. Bez tohoto nastavení vznikaly problémy například u písmene „M”, kde ostrý úhel způsobil extrémní protažení obrysu. Při některých hodnotách a několikanásobném zvětšení cesty od středu však vznikaly zajímavé tvary obrysů, které jsme se nakonec rozhodli použít.

Uživateli je tedy nabídnuto několik režimů obrysu, které se implementačně od běžného obrysu liší jiným nastavením typu rohů cesty, jemností cesty při převodu z vektorového fontu, atd. U nadpisu je navíc pro pestřejší vzhled přidán druhý (vnitřní) obrys obyčejného tvaru.

Výplň textu a obrysů je nastavitelná barevným přechodem, vyskytl se zde však problém s různě velkými mezerami mezi fonty, které znemožnily vykreslit barevný přechod správně. Bylo tedy potřeba vykreslit každý řádek nezávisle na ostatních. Aby se nemusely propočítávat správné souřadnice všech řádků, řešili jsme tuto situaci implementačně tak, že vykreslujeme stále stejný několikařádkový textový řetězec, kde však v každém kroku nahradíme obsah všech řádků, které právě nevykreslujeme, prázdným textovým řetězcem.

(Obrázek 24) bílý obrys nadpisu a dodatečný černý obrys s ozvláštněným tvarem

5.4 Filtrace obrazu

Při návrhu grafických filtrů jsme nejprve zvažovali možnosti využití některého profesionálního programu, jehož filtry bychom mohli využít. Zjistili jsme například, že je možné programově ovládat (programováním pomocí objektového modelu) spuštěnou instanci programu ***Adobe Photoshop*** (to je v jeho nejnovějších verzích umožněno). To by nám ideálně zajistilo použití filtrů přesně tak, jak jsme je navrhli. Postup je popsán na těchto stránkách[1].

Bohužel jsme neměli možnost na školní server tento program umístit a navíc skupina uživatelů, kteří by mohli tuto funkčnost využít, by byla malá. Další možnost, kterou jsme zkoumali, bylo využití grafických filtrů, využívaných v programu ***Adobe Photoshop*** ve formě zásuvných modulů. Jsou jimi soubory s příponou ***8bf***, ve skutečnosti jsou to ale dynamické knihovny typu ***dll***. Je možné je po dodání vhodných struktur a bitmap z kteréhokoliv programu zavolat. Bohužel se nám nepodařilo tuto funkčnost plně implementovat, protože mnoho nových verzí filtrů již nepodporuje rozhraní, které je popsáno ve starších verzích vývojářského nástroje, jehož nové verze nejsou pro běžné uživatele k dispozici. V současné době tak tuto možnost zčásti využívá například grafický software ***GIMP***[2] nebo prohlížeč ***IrfanView***[3], oba ale podporují pouze některé staré verze filtrů, které pro nás nemají praktický význam. Jak jsme tedy již od počátku tušili, nezbylo nám než implementovat vlastní filtry a pokusit se jimi napodobit postupy, které jsme navrhli v předcházející kapitole.

V naší aplikaci používáme pro filtraci obrazu funkce pro změnu barev, úrovní histogramu, vytvoření polotónového vzoru a jeho následnou aplikaci na obrázek. Námi implementované filtry pro změnu barev fungují převážně na principu převodu na obrázek s potlačenou barevnou sytostí a s ponechaným odstínem určité barvy.

Filtry pracují v barevném modelu ***HSV***, je proto potřeba je do tohoto modelu převést. Pro každý nový obrázek se při načtení provede automatické vyrovnání histogramu a v této formě je obrázek uložen pro další zpracování. Uživatelem nastavitelné hodnoty normalizace histogramu tedy nejsou v absolutních jednotkách, ale relativní vůči obrázku s již ekvalizovaným histogramem. Tím se dá dosáhnout vyšší pravděpodobnosti, že budou mít obrázky navzájem vůči sobě stejný kontrast. Všechny filtry, kromě vytvoření polotónového vzoru, jsou bodové operace. Vytvořili jsme pro ně obecnou metodu ***Filters_ApplyFilter***, která na každý obrazový bod vstupního obrazu aplikuje zadanou vstupní filtrovací funkci. Tyto filtrovací funkce pracují s ukazatelem na tříbajtové pole hodnot jednoho pixelu (pro zdrojové obrázky používáme 24bitovou hloubku barev). Pro naši aplikaci jsme implementovali následující filtrovací funkce.

[1] Deurbrouck J.: *Accessing the Photoshop CS Interface via COM*. http://www.pcpix.com/Photoshop/ [duben, 2006]

[2] http://www.gimp.org/

[3] http://www.irfanview.com/

5.4.1 Filtry pro změnu barev

Filters_RGBtoHSV – převede každý obrazový bod v zadané bitmapě z barevného prostoru ***RGB*** do barevného prostoru ***HSV***. Převod je proveden pomocí schématu (1). Jako hodnotu pro ***H*** „nedefinováno“ jsme zvolili číslo 255.

$$H = \begin{cases} nedefinováno, & pokud\ MAX = MIN \\ 60° \times \dfrac{G-B}{MAX-MIN} + 0°, & pokud\ MAX = R \wedge G \geq B \\ 60° \times \dfrac{G-B}{MAX-MIN} + 360°, & pokud\ MAX = R \wedge G < B \\ 60° \times \dfrac{B-R}{MAX-MIN} + 120°, & pokud\ MAX = G \\ 60° \times \dfrac{R-G}{MAX-MIN} + 240°, & pokud\ MAX = B \end{cases} \qquad S = \begin{cases} 0, & pokud\ MAX = 0 \\ 1 - \dfrac{MIN}{MAX}, & jinak \end{cases} \qquad V = MAX$$

(1) schéma převodu z modelu ***RGB*** na ***HSV***[1]

Filters_HSVtoRGB – inverzní funkce k výše popsané funkci. Obousměrný převod jsme ale přizpůsobili tak, aby zachoval nedefinovanou hodnotu v oblasti odstínu.

$$H_i = \left\lfloor \frac{H}{60} \right\rfloor \bmod 6 \qquad f = \frac{H}{60} - H_i \qquad p = V \cdot (1 - S) \qquad q = V \cdot (1 - f \cdot S) \qquad t = V \cdot (1 - (1 - f) \cdot S)$$

$$\begin{aligned} &pokud\ H_i = 0 \rightarrow && R = V,\ G = t,\ B = p \\ &pokud\ H_i = 1 \rightarrow && R = q,\ G = V,\ B = p \\ &pokud\ H_i = 2 \rightarrow && R = p,\ G = V,\ B = t \\ &pokud\ H_i = 3 \rightarrow && R = p,\ G = q,\ B = V \\ &pokud\ H_i = 4 \rightarrow && R = t,\ G = p,\ B = V \\ &pokud\ H_i = 5 \rightarrow && R = V,\ G = p,\ B = q \end{aligned}$$

(2) schéma převodu z modelu ***HSV*** do ***RGB***[1]

Filters_Levels – spočítá výsledný jas obrazového bodu dle uživatelem nastavitelných hodnot vstupních úrovní histogramu. Pracuje v režimu ***HSV*** a ovlivňuje pouze jasovou složku, čímž dodává obrazům při vyšším kontrastu pastelový vzhled. V níže uvedeném vzorci (3) pro normalizaci obrazu, který tato funkce implementuje, označuje ***Vin*** vstupní úroveň jasu, ***Vout*** výstupní, ***B*** je hodnota vstupní úrovně černé barvy a ***W*** bílé. Hodnoty rozmezí výstupních hodnot jsou v našem případě číselné konstanty 255 a 0.

$$V_{out} = (V_{in} - B) \cdot \left(\frac{255 - 0}{W - B} \right) + 0$$

(3) vzorec pro normalizaci úrovní obrazu doplněný o konstanty 255 a 0

[1] Wikipedia, the free encyclopedia: *HSV color space*. http://en.wikipedia.org/wiki/HSV_color_space [říjen, 2006]

Filters_BWinHSV – nastaví hodnotu barevné sytosti na hodnotu zadanou uživatelem, výsledkem je tedy obrázek s potlačenou barevnou sytostí.

Filters_MakeHueHistogram – nalezne nejčastěji se vyskytující (dominantní) barevný odstín.

Filters_BWinHSVkeepDominant – potlačí barevnou sytost obrázku s výjimkou dominantního odstínu, nalezeného pro každé políčko zvlášť. Práh tolerance, kdy se ještě daný barevný odstín plně ponechává, může uživatel měnit. S dále narůstající odchylkou již klesá saturace barev lineárně dle uživatelem nastavitelné hodnoty strmosti tolerance. Tím je dosaženo plynulého přechodu mezi zabarvenými a méně zabarvenými (dne nastavení celkové sytosti) částmi obrazu.

Filters_BWinHSVkeepDominantDefined – obdoba výše popsané funkce. Odstín dominantní barvy se přebarví na uživatelem definovaný odstín. Uživatel má možnost buď použít jeden konkrétní odstín, nebo část spektra, které odpovídá rozsahu spektra dominantní barvy.

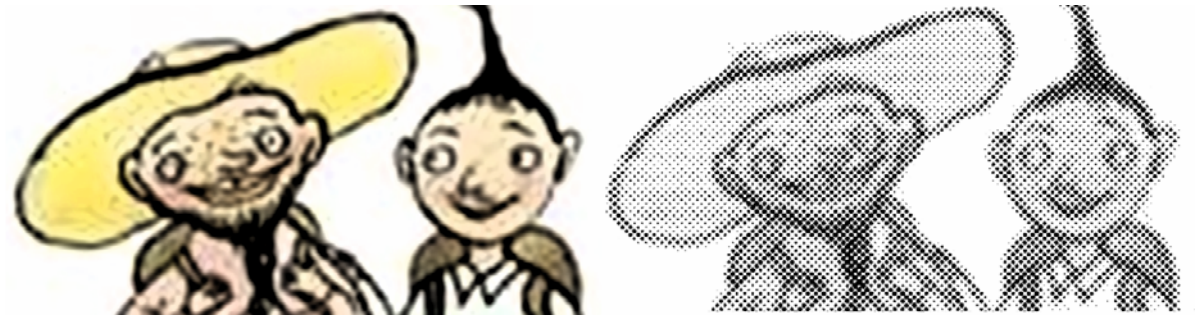
Filters_BWinHSVkeepDefined – obdoba výše popsané funkce. Ponechá se uživatelem nastavený odstín barvy. Tato funkce také umožňuje ponechat buď jen jeden konkrétní odstín, nebo část spektra, která odpovídá toleranci.

5.4.2 Filtry pro polotónový překryv

Filters_MakeNewHalftone – tato funkce vytvoří ze vstupní bitmapy černobílý polotónový vzor a uloží jej do pole bitů, které v našem programovacím prostředí reprezentuje třída ***BitArray***. Při implementaci jsme zcela vycházeli z online zdroje[1]. Bylo potřeba algoritmus přepsat do jazyka ***C#***, výrazně jej optimalizovat a opravit některé drobné chyby v jeho návrhu.

Algoritmus vychází ze třídy tzv. ***Clustered-Dot Ordered Dither*** algoritmů. Jeho postup je následující. Nejprve je celý podklad vyplněn bílou barvou. Obrazové body vstupního obrazu se sloučí do bloků (jejich rozměr je uživatelem definovaný a vyjadřuje velikost výsledných teček), které jsou pravidelně uspořádané v mřížce. V těchto blocích se pro každý vstupní pixel vypočítá obrazová funkce (dle uživatelem zadané hodnotě úhlu řádkového rozkladu), jejíž výsledky se uspořádají a následně prahují s vypočtenou střední hodnotou jasu v celém bloku. Výsledkem je poté index, který udává tvar (velikost) černé tečky uvnitř bloku.

[1] Lau D. L.: *Digital halftoning*. zdrojový kód, verze z roku 9. 6. 1998.
http://www.engr.uky.edu/~dllau/Halftone/HtmlFiles/Software/HTMLfiles/cdod.html



(Obrázek 25) obrázek[1] a jeho polotónový vzor

Filters_Halftone_Multiply – tato funkce překryje bitmapu polotónovým vzorem. Překryv je uskutečněn pomocí operace násobení dvou barev navzájem (v grafických programech se označuje jako operace barevného násobení). Jelikož má uživatel možnost zadat tmavou i světlou polotónovou barvu, násobí se hodnoty barev mezi sebou v každém ze tří barevných kanálů; mezivýsledek se s uživatelem nastavitelnou mírou kombinuje s originálním obrazem.

Tato funkce, díky množství operací násobení a dělení pro každý obrazový bod, vyžadovala optimalizaci především ve formě předpočítaných koeficientů dle uživatelem zvolených hodnot. Díky nim je výpočet redukován na jedno násobení a jeden bitový posun pro jeden pixel v každém barevném kanálu.

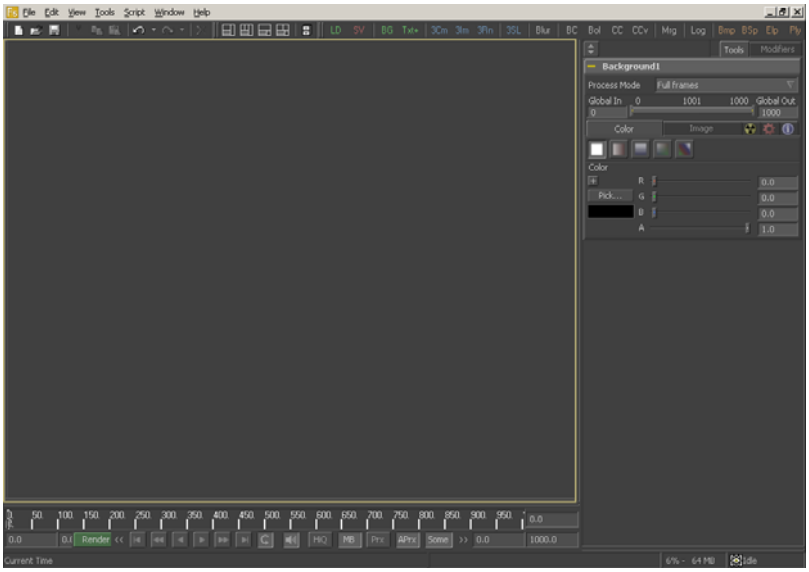
(Obrázek 26) původní obrázek[2], jeho obarvený polotónový vzor, překryv aplikovaný operací ***multiply***

[1] Skála F.: *Velké putování vlase a brady*. Arbor Vitae, 2007

[2] http://goodcomics.blogspot.com/

5.5 Grafické uživatelské rozhraní

Při implementaci grafického uživatelského rozhraní spolu s jeho ovládáním jsme se inspirovali některými profesionálními programy pro pokročilou práci s obrazem a videem. Jedním z nich je program pro pokročilou editaci obrazu, ***Fusion***[1].

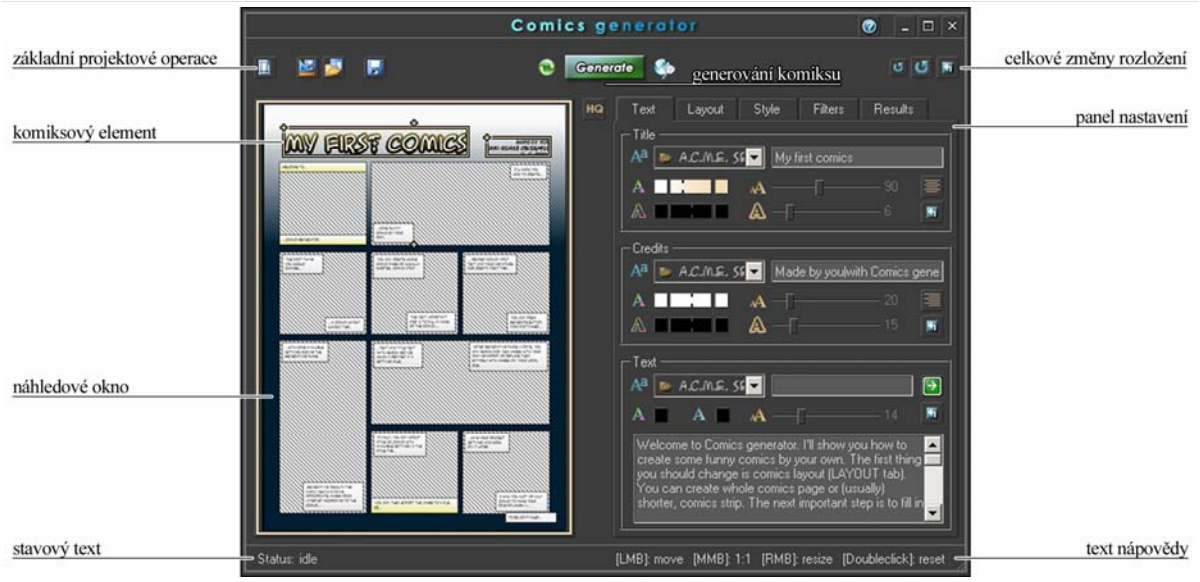


(Obrázek 27) vzhled grafického uživatelského rozhraní programu ***Fusion***

Z tohoto programu jsme převzali některé společné rysy. Většinu obrazovky zabírá okno s náhledem komiksu, ve kterém lze pomocí myši s náhledem interaktivně pohybovat a měnit jeho velikost. Dále v něm lze pohybovat s některými komiksovými elementy. Na pravé straně se nachází panel nastavení, který pomocí záložek odděluje skupiny nastavení do logických celků. Ve spodní části programu se nachází stavový řádek a krátká textová nápověda ke každému ovládacímu prvku, na kterém se právě nachází kurzor myši. V horní části se nachází tlačítka pro základní práci s projektem a generování komiksu. Podobné rozvržení respektuje většina výrobců profesionálního grafického software.

Jelikož má většina standardních ovládacích prvků v našem programovacím prostředí omezené možnosti nastavení chování a vzhledu, využili jsme možnost vytvořit si vlastní ovládací prvky pomocí dědění třídy ***UserControl***. To nám nejenom zajistilo nezbytnou funkčnost některých ovládacích prvků, které nemají v našem programovacím prostředí ekvivalent, ale umožnilo nám to též sjednotit vizuální vzhled všech ovládacích prvků pomocí nové implementace metody ***OnPaint***, která vykresluje ovládací prvek na okno aplikace. Uvádíme nejzajímavější ovládací prvky, které jsme implementovali.

[1] http://www.eyeonline.com



(Obrázek 28) přehled prvků grafického uživatelského rozhraní

5.5.1 Ovládací prvek náhledu komiksu

Ovládací prvek ***ViewPicture*** slouží k náhledu libovolného obrázku ve formě bitmapy. S obrázkem je možné levým tlačítkem myši pohybovat, pravým tlačítkem měnit jeho zvětšení, prostředním zvětšit na skutečnou velikost v pixelech a nakonec je možné jej dvojklikem vycentrovat na celou šířku či výšku ovládacího prvku. Využíváme jej i k dalším účelům, jako je interaktivní pohybování s komiksovými elementy. Bylo proto potřeba implementovat mechanismus pro vzájemný převod mezi dvěma souřadnicovými systémy – souřadnicový systém obrazovky a zobrazované bitmapy.

Veškeré komiksové elementy, se kterými lze interaktivně hýbat, jsou reprezentovány objektem typu ***Offset***, který v sobě nese informace o původním (automatickém) umístění, odchylce od tohoto umístění a svých rozměrech. Ovládací prvek využívá tyto objekty k orámování elementů, se kterými lze pohybovat. Při pohybu myši se zjišťuje, zda souřadnice kurzoru není obsažena v některém z těchto objektů. Pokud ano, vysvítí se příslušný komiksový element. Při stisku příslušného tlačítka se spolu s pohybem myši mění jeho pozice.

(Obrázek 29) výřez náhledového okna s pohyblivými prvky

5.5.2 Ukládání a načtení projektového souboru

Hlavním obecným požadavkem při ukládání a načítání projektových souborů je, aby projekt byl při načtení ve stejném stavu, jako byl při jeho uložení. V naší implementaci je toho dosaženo tím, že ukládáme veškeré uživatelem nastavitelné hodnoty, uložené ve statické třídě ***Settings***. Naše programovací prostředí disponuje užitečnou možností odkázat se na programovou proměnnou pomocí textového řetězce a naopak. Díky tomu není potřeba ukládat každou proměnnou zvlášť a v pevně daném pořadí.

Cyklus načtení projektu tedy funguje tak, že ze souboru načte řádek, podle textového řetězce v první části řádku (před znakem „:“) zjistí, o kterou proměnnou se jedná, a uloží do ní hodnotu z druhé části řádku. K tomu je potřeba řetězec hodnoty převést na správný typ. Výčtové typy lze v našem programovacím prostředí také převádět na jejich textovou reprezentaci a zpět. Odlišný postup je potřeba použít u typů ***FontFamily*** (rodina fontů), ***Color*** (barva) a námi definovaného typu ***Offset***. U rodiny fontů se ukládá její jméno a při načtení se program pokusí najít tento font v seznamu dostupných fontů. Pokud není nalezen, použije se standardní systémový font. U barvy se pro převod na řetězec využívá metody statické třídy ***ColorTranslator***. Hodnoty třídy ***Offset*** se zpracovávají po částech (jsou oddělené čárkou). Do projektového souboru se též ukládá semínko pro vytvoření pseudonáhodné posloupnosti čísel.

Druhá, volitelná část konfiguračního souboru, obsahuje pro každé komiksové políčko (a pro pozadí) relativní odkazy na obrázkové soubory, jejich změny pozice a velikosti (***Offset***) a nalezená klíčová slova. Obrázky se ukládají do podadresáře, jehož název končí názvem projektového souboru. Názvy souborů jsou jedinečně pozměněné, aby nedošlo k přepsání jiných obrázků při ukládání projektu (mnoho obrázků na internetu má shodný název souboru). Spolu s obrázky se ukládají také jejich polotónové vzory (s příponou ***htp***), aby načtení projektu trvalo kratší dobu. Přitom je potřeba převést je z binární podoby na reprezentaci v bajtech a při ukládání naopak.

5.5.3 Panel nastavení

Pro lepší vizuální oddělení jsou uživatelem nastavitelné hodnoty rozděleny do několika logických kategorií, které korespondují se záložkami v panelu nastavení. Protože typický ovládací prvek ***TabControl*** v našem programovacím prostředí neposkytuje úplnou kontrolu nad vzhledem a chováním, implementovali jsme vlastní panel s pomocí zdrojového kódu z tohoto zdroje[1]. Dále jsme pro lepší orientaci v nastavení vytvořili sadu grafických ikon, které mají vizuálně napovědět funkci daného ovládacího prvku.

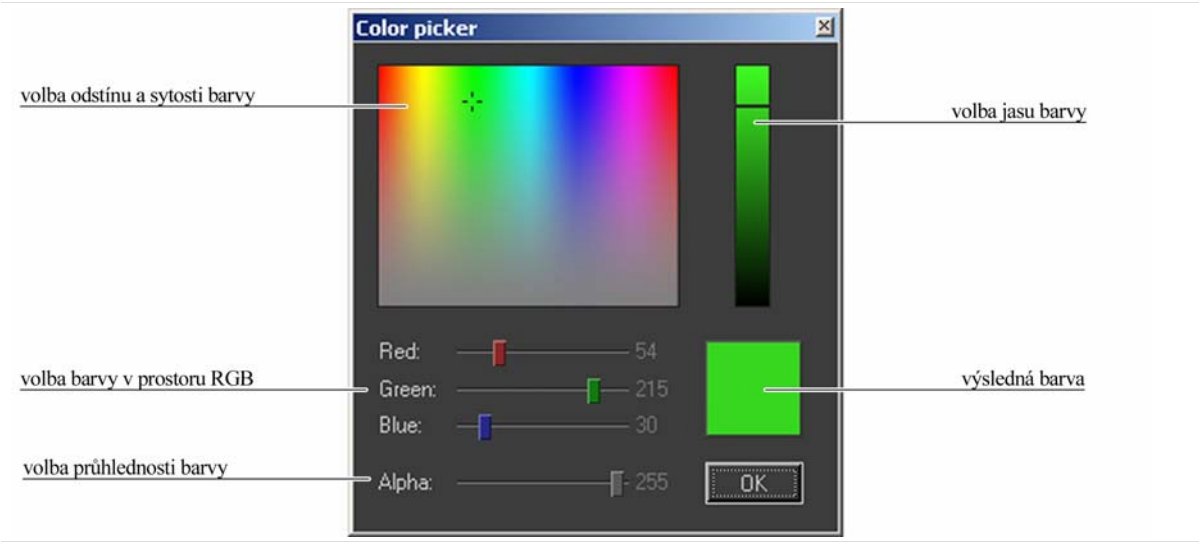
[1] Doherty M.: *DotNet tips*. http://homepage.ntlworld.com/mdaudi100/alternate/controls.html [březen, 2007]

5.5.4 Panel výsledků

V záložce panelu výsledů (***Results***) se nachází ovládací prvky pro nastavení dodatečných vyhledávacích klíčových slov, přepínaní volby slovníku a obrázkové vyhledávací služby. Především se zde ale nachází panel, ve kterém je možné manuálně zadat klíčová slova a obrázky nezávisle pro každé políčko. Nachází se zde právě tolik dynamicky generovaných ovládacích prvků, kolik je aktuálně v komiksu políček. Dále je zde možné nastavit aktuální políčko, jehož obrázek je možné interaktivně přesouvat. Kdyby byly dostupné všechny, bylo by náhledové okno nepřehledné.

5.5.5 Dialogové okno pro výběr barvy

Pro výběr uživatelem definovatelné barvy jsme implementovali vlastní dialogové okno, které umožňuje vybrat barvu jak v barevném prostoru ***RGB***, tak i ***HSV***. Navíc je zde možnost nastavit průhlednost barvy (***Alpha***). Pro správné pozice a hodnoty ovládacích prvků jsme implementovali shodné metody pro převod mezi ***RGB*** a ***HSV***, které používáme při filtraci obrázků. Naše programovací prostředí totiž sice umožňuje zjištění ***HSV*** komponent pro zadanou barvu, bohužel ale funguje na jiném principu převodu než naše metody (například pro velmi světlou barvu dávají hodnotu saturace blízkou maximu).



(Obrázek 30) dialogové okno pro výběr barvy

5.5.6 Ovládací prvek pro výběr barevného přechodu

Aby se daly vlastnosti uživatelem nastavitelných barevných přechodů měnit přímo v okně aplikace, implementovali jsme ovládací prvek, který v sobě obsahuje výběr počáteční barvy, koncové barvy a pozice středu barevného přechodu. Dvojklikem na střed barevného přechodu je možné zvolit jednotnou barvu (neboli nastavit oba konce barevného přechodu na shodnou barvu).

5.6 Online klientské rozhraní

Pro ukázku funkčnosti serverového režimu běhu programu jsme vytvořili jednoduché stránky v jazyce ***PHP*** a ***HTML*** na internetové adrese[1], která nám byla na škole poskytnuta. Stránky nabízí možnost zadání nadpisu, titulků a komiksového textu. Ostatní parametry jsme nastavili v projektovém souboru ***netsettings_default.cmx***, který je programem načten jako šablona. Vytvořili jsme jej tak, že jsme v programu vytvořili komiks a uložili projektový soubor pod tímto jménem.

Do dalšího souboru ***netsettings.cmx*** se ukládají již hodnoty, získané od uživatele ze stránky; přepíšou tím hodnoty načtené ze šablony. Jelikož ***PHP*** server, který nám byl k dispozici, má nastavenou maximální dobu zpracování spustitelného souboru na 30 vteřin, bylo potřeba nastavit šablonu na méně políček s nižším rozlišením obrazu, aby bylo možné za tuto dobu komiks vygenerovat. Programu je možné zadat na příkazové řádce název výstupního grafického souboru. Naše implementace online rozhraní zatím ukládá obrázek do souboru ***img.jpg***, ale v pokročilejším online rozhraní by bylo vhodné ukládat jej s unikátním jménem, aby se soubory nepřemazávaly.

5.7 Přehled uživatelem nastavitelných hodnot

Nastavení parametrů textu

- vlastnosti fontu (styl, velikost, barva) nadpisu, titulků, textu a alternativního textu
- pozice a velikosti nadpisu a titulků (interaktivně)
- vlastnosti obrysu (velikost, barevný přechod) nadpisu a titulků
- horizontální zarovnání textu nadpisu a titulků při více řádcích
- textový řetězec nadpisu, titulků a textu

Nastavení rozvržení stránky

- velikost obrázku v obrazových bodech (přednastavené hodnoty i manuální nastavení)
- rozměr mřížky, která slouží jako základ pro generování velikosti a umístění políček
- odsazení mřížky od okrajů komiksu
- šířka a výška žlábku a její variabilita
- náhodnost variace umístění políček vzhledem k mřížce
- variabilita velikosti komiksových políček
- míra zaplnění mřížky komiksovými políčky
- vzdálenost popisků od okraje políčka a její variabilita
- zarovnání popisků (vlevo nahoře až vpravo dole)
- pravděpodobnost výskytu alternativních popisků

[1] http://ilert.fit.vutbr.cz/dotnet/comics.php

Nastavení vizuálního stylu

- barevný přechod pozadí (interaktivně jeho začátek a konec)
- tloušťka a barva orámování komiksu
- mód filtrace obrázku na pozadí
- hranice průhlednosti vrchního a spodního okraje
- styl dodatečného obrysu nadpisu, jeho barevný přechod, velikost (pozice interaktivně)
- tloušťka a barva orámování políček a obou stylů popisků
- barevný přechod pozadí obou stylů popisků
- odsazení textu od kraje popisků

Nastavení obrazových filtrů

- styl obarvení (nezměněný, černobílý, černobílý s nalezeným dominantním odstínem, černobílý se zvoleným odstínem, psychedelický se zvoleným posuvem, atd.)
- parametry tolerance a útlumu tolerance pro zvolený barevný odstín
- parametry pro změnu vstupních úrovní jasu (pro nastavení kontrastu)
- celkový útlum barevné sytosti
- styl aplikace polotónového vzoru (překryv, dvoubarevný, žádný)
- velikost polotónových obrazových teček
- úhel polotónového vzoru
- míra prolnutí původního obrázku s obrázkem po aplikaci polotónového vzoru

Dodatečná změna nalezených obrázků

- nastavení dodatečných klíčových slov pro nalezení obrázků
- změna online slovníku
- změna obrázkové vyhledávací služby
- manuální zadání klíčových slov pro pozadí a každé políčko zvlášť
- vygenerování nové adresy obrázku ze zadaných klíčových slov
- manuální zadání adresy obrázku pro pozadí a každé políčko zvlášť
- smazání obrázku

6 Zhodnocení a výsledky

6.1 Grafické uživatelské rozhraní

Program jsme nabídli k vyzkoušení několika uživatelům, jejichž poznatky zde stručně shrneme. Program vzhledem působil dobře, ale uživatelé, kteří nepracovali s grafickým software nebo předem neznali ovládání programu, se v něm špatně orientovali. To i přesto, že k programu je dodána nápověda a jsou implementovány nápovědné popisky ke každému ovládacímu prvku, bohužel však jen v anglickém jazyce. Jeden ze způsobů, jak aplikaci plně zpřístupnit širokému okruhu uživatelů bez omezení její funkčnosti, by bylo skrýt mnohé z nabídek a ponechat jen několik nejdůležitějších.

K tomu vybízí možnost vytvořit online klientské rozhraní, kde se budou nacházet jen vybrané ovládací prvky, jako je tomu i v případě námi implementované testovací internetové stránky. Ta však v tuto chvíli nabízí nastavení pouze tří hodnot (nadpis, titulky a text komiksu).

Vyplývá z toho také skutečnost, že většina běžných uživatelů se orientuje pouze v zažitém standardním vzhledu aplikací pro operační systém ***Windows*** a pokud mají ovládat program s jiným stylem grafického uživatelského rozhraní, je pro ně nesnadné se v krátkém čase adaptovat.

Přestože jsme použili množství optimalizací pro vykreslení jednotlivých vrstev komiksu, není možné v naší aplikaci interaktivně vykreslovat komiks v původním obrazovém rozlišení a ve vysoké kvalitě zobrazení tak, jak to profesionální grafické aplikace typu ***Adobe Photoshop*** běžně umožňují. ***GDI+*** je pro takovou funkcionalitu příliš pomalé. Podobné aplikace je potřeba vyvíjet buď s pomocí ***GDI***, ***OpenGL*** nebo ***DirectX***. V nízké kvalitě zobrazení je však možné na dnešních běžných počítačích s komiksem interaktivně pracovat. Čím větší je však počet použitých políček, tím pracuje aplikace při interaktivních změnách pomaleji.

Při testování jsme zjistili další zajímavou skutečnost. Jednotky pixelů, používané v ***GDI+*** při nastavení pozice či velikosti grafických elementů, ve skutečnosti odpovídají skutečným pixelům pouze tehdy, když má operační systém ***Windows*** nastavené rozlišení monitoru 96 dpi (tak tomu zatím ve většině případů je). Některé novější monitory již však mají přednastavenou větší hodnotu ***DPI*** a tak se jednotky, již zkompilované v programu, teprve za běhu převádějí na skutečné pixely. Zároveň se tento přepočet ale liší pro velikost fontů, která je zadána pomocí jiných jednotek. To v mnoha dnes dostupných programech způsobuje nejrůznější chyby v rozměrech ovládacích prvků, velikostí fontů, atd. My jsme vypnutím parametru ***AutoScaleMode*** zajistili, aby byly všechny ovládací prvky dostupné při jakémkoliv nastavení ***DPI***, přestože aplikace může vizuálně vypadat jinak, než jak byla navržena. Jak vytvořit aplikaci, která je připravena na toto chování a která vykresluje ovládací prvky zcela správně, je popsáno ve článku[1] na ***MSDN***.

[1] Microsoft: *Developing DPI-Aware Applications*. http://msdn2.microsoft.com/en-us/library/ms838191.aspx [únor, 2007]

6.2 Analýza klíčových slov

V této kapitole zhodnotíme výsledky analýzy vstupního textu a nalezených klíčových slov. Na několika anglických slovíčkách jsme testovali výsledky, dle použitého slovníku. Bohužel oba slovníky, které jsou v aplikaci standardně dostupné, mají občas nevhodně zařazená některá slova. Například anglické slovo „how“ se v námi použitém tezauru ani nenachází, zato v online slovníku ano. (Tabulka 2) ukazuje bodové ohodnocení několika anglických slovíček, které reprezentují různé slovní druhy. V řádcích je bodové ohodnocení slov při daném slovníku. Zelenou barvou je znázorněn nejlepší výsledek pro dané slovo a červenou barvou nejhorší, podle toho, jak body korespondují se subjektivní informační hodnotou daného slova. Jedná se o subjektivní hodnocení správnosti bodování, nelze tedy tuto tabulku brát jako výsledek nějakého měření. Z tabulky je patrné, že použití slovníku či tezauru, ač zpomaluje analýzu klíčových slov, výrazně zlepšuje kvalitativní výsledek analýzy. Český text jsme do tabulky nezahrnuli, protože jsme pro něj neměli k dispozici online slovník, který by zobrazoval slovní druhy.

	čas (s)	welcome	first	comics	generator	you	how
slovník[1]	120	30	35	40	80	10	10
tezaurus[2]	90	45	35	65	80	10	50
bez	20	70	60	65	80	55	50

(Tabulka 2) bodové ohodnocení ukázkových anglických slovíček

Pro testování zcela odlišného jazyka jsme zvolili japonštinu. Ta pro naše účely reprezentuje jazyk, který používá pro vyjádření významu jednotlivé znaky, místo slov, oddělených mezerou. Program text rozdělil správně až po přepsání konfiguračního souboru, jelikož japonština obsahuje jiná interpunkční znaménka. Program poté již správně odděloval jednotlivé věty. Rozdělení slov je však v tomto jazyce, jak jsme již dříve zmínili, náročná úloha. Grafické uživatelské rozhraní v tomto jazyce pracovalo správně (Obrázek 31). Bodové ohodnocení však neuplatnilo algoritmus vyhledávání v online slovníku. V japonštině nelze dále uplatnit náš algoritmus na odhadnutí informační hodnoty daného slova podle jeho délky. Jelikož se tedy hodnotil počet nálezů celé věty, analýza klíčového slova dala nespolehlivé výsledky a nalezené obrázky byly téměř zcela náhodné. Jeden z testovacích japonských komiksů se nachází v příloze této práce.

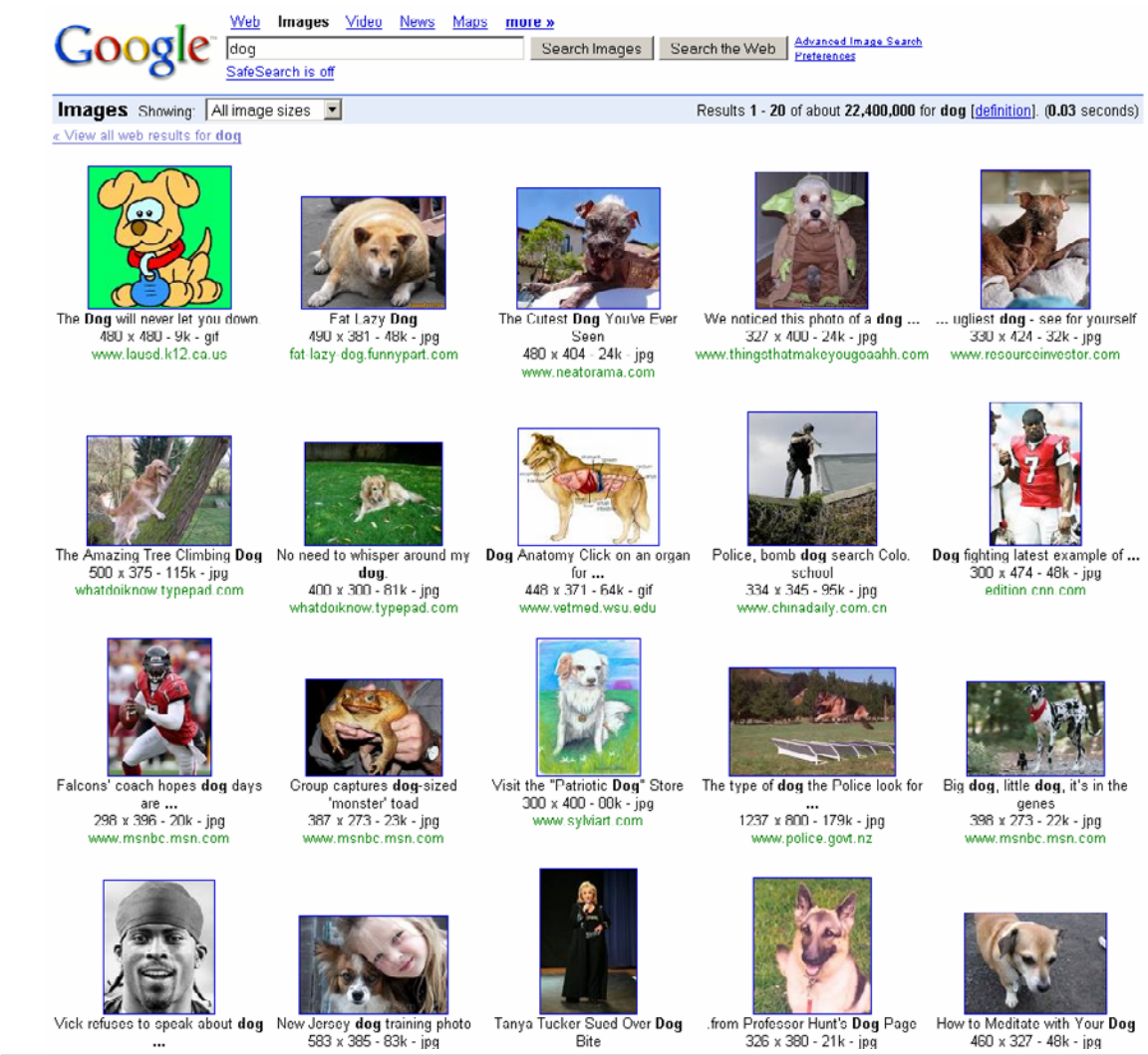
(Obrázek 31) použití japonského textového řetězce v grafickém uživatelském rozhraní

[1] http://dictionary.reference.com

[2] http://thesaurus.reference.com

6.3 Nalezení obrázků vyhledávací službou

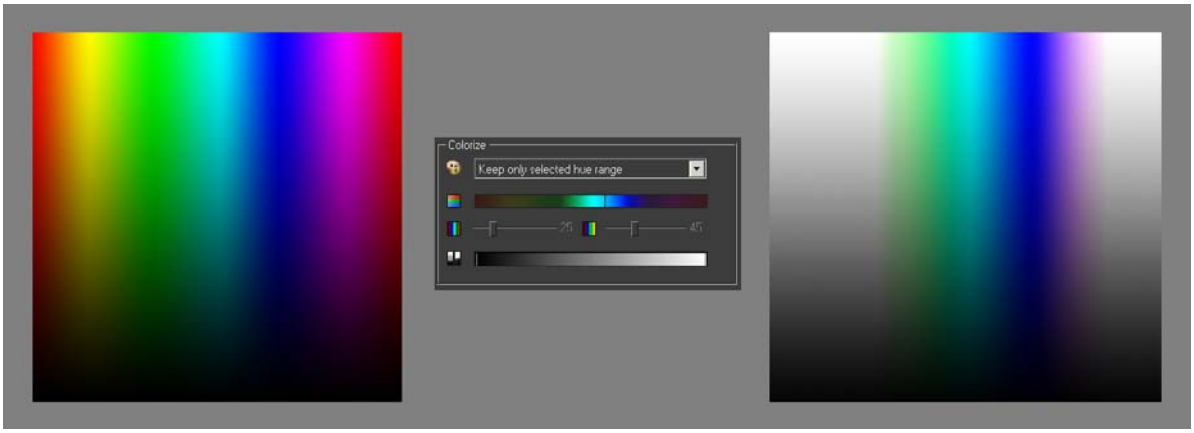
Po vyhodnocení adekvátnosti nalezených klíčových slov, která reprezentují dané komiksové políčko, zbývá zhodnotit adekvátnost nalezení obrázku z těchto klíčových slov. Pomocí dnes dostupných vyhledávacích služeb bohužel není možné najít se stoprocentní jistotou obrázek, který přesně odpovídá danému klíčovému slovu. Proto jsme již zpočátku tvořili tento program za účelem vytváření humorných až absurdních výsledků a umožnili jsme v případě potřeby kterýkoliv obrázek nahradit jiným. Procentuální poměr správně nalezených výsledků je přímo závislý na charakteristice dané vyhledávací služby. Některá velmi často užívaná slova, jako například anglické slovo „dog“, bývají nalezena až s 90% úspěšností (Obrázek 32), některá však s mnohem menší. V jiných, méně často užívaných jazycích, se tento poměr blíží až jednotkám procent.



(Obrázek 32) nalezené obrázky pro slovo „dog“ pomocí ***Google images***

6.4 Vyhodnocení grafických filtrů

Pro testování správnosti našich grafických algoritmů jsme používali mimo jiné následující obrázky. (Obrázek 33) ukazuje testovací barevné spektrum a výsledný obrázek po aplikaci filtru, který ponechá jen určitý rozsah barevných odstínů.



(Obrázek 33) barevné spektrum (vlevo) a aplikace filtru potlačení barevné sytosti (vpravo) s ponecháním zvoleného barevného spektra (uprostřed)

(Obrázek 34) vyobrazuje tři ukázkové vstupní obrázky. Vlevo je původní obrázek, uprostřed jeho digitálně podexponovaná a napravo přeexponovaná varianta. Pod těmito obrázky se nachází výsledné obrázky s aplikovanou normalizací jasových úrovní. Zároveň je zde demonstrována aplikace filtru záměny dominantního barevného odstínu za uživatelem zvolený barevný odstín.

(Obrázek 34) normalizace jasových úrovní a aplikace filtru potlačení barevné sytosti s ponecháním fialové barvy

Dále jsme porovnali dva vybrané, původně navržené filtry, a postupy s výslednou implementací našich vlastních filtrů. (Obrázek 35) ukazuje navržený postup polotónování (nalevo) a obrázek filtrovaný pomocí implementovaného postupu polotónování s překryvem (napravo).

(Obrázek 35) porovnání navrženého a implementovaného postupu polotónování

(Obrázek 36) v podobném duchu ukazuje porovnání navrženého a implementovaného postupu prahování a útlumu barevné sytosti s ponecháním zvoleného barevného odstínu. V tomto případě jsme se pomocí naší implementace přiblížili kýženému výsledku o něco více.

(Obrázek 36) porovnání navrženého a implementovaného postupu prahování a útlumu barevné sytosti

7 Závěr

7.1 Shrnutí

Implementace programu pro automatickou tvorbu komiksové stránky byla úspěšně zakončena. Jednoznačně nám v tom pomohlo robustní programovací prostředí ***.NET Framework***, bez kterého bychom některé postupy museli zdlouhavě a obtížně implementovat. V rámci práce se nám podařilo implementovat i většinu úkolů, které jsme si zvolili jako dodatečné. Vyzkoušeli jsme si postupy programování z oblasti grafických filtrů, designu a interaktivní práce s obrázky. Tyto zkušenosti jsou dobrým základem pro tvorbu profesionálních grafických programů.

V současném stavu splňuje naše aplikace podmínky, které jsme na ni v zadání kladli. Přesto by však mohla být ještě více optimalizována pro interaktivní grafické manipulace. Strukturovali jsme implementaci aplikace tak, aby bylo možné vytvořit některá níže navržená rozšíření a v práci tak dále pokračovat.

7.2 Budoucí rozšíření

Vhodným rozšířením a pokračováním této práce by byla implementace pokročilého internetového rozhraní, které by naši aplikaci využívalo a obohatilo ji o širší možnosti nastavení vzhledu komiksu, přehledu již vytvořených komiksů od různých autorů, jejich hodnocení, žebříčky hodnocení a další zajímavé funkce, které však přesahují rámec této práce. Pro další rozšíření aplikace o toto rozhraní uvádíme v příloze nastavení aplikace pro běh v serverovém režimu.

Druhé možné rozšíření práce by bylo vytvoření dalších grafických filtrů, produkujících z obrázku malbu nebo jinou zajímavou vizuální variaci. Pokročilé grafické filtry pro transformaci fotografie na obrázek jsou v profesionální oblasti velmi ceněné a není jich zatím příliš mnoho. Práce na toto téma může sloužit jako dobrý základ v oblasti profesionálního zpracování obrazu.

8 Seznam použitých zdrojů

Zdroje ilustrací

1) O' Malley B. L.: ***Lost at Sea***. Oni Press, 2006
2) Frazetta F.: ***Al Capp's Li'l Abner: The Frazetta Years, Volume 1***. Dark Horse, 2003
3) Duggan G., Posehn B.: ***The last Christmas***. Image Comics, 2006
4) Davis J.: ***Garfield Classics: Vol 14***. Ravette Publishing Ltd, 2004
5) McCay W.: ***Little Sammy Sneeze***. Checker Book Publishing Group, 2004
6) Fawstin B.: ***Table for one***. Mainspring Comics, 2004
7) Parker S., Alexander J.: ***Across the Pond Presents #2***. Across the Pond Studios, LLC, 2004
8) Marvel Comics: ***Invaders meet Frankenstein #31***, Marvel Comics, 1978
9) Mackie H.: ***Spectacular Spider-Man #263***, Marvel Comics, 1998
10) Anderson E. A., Tembley M.: ***Sam Noir samurai detective,*** Image Comics, 2007
11) Lee S., Roy T.: Spider-Man: ***Saga of the Sandman***. Marvel Comics, 2007
12) Straczynski J.M., Youngquist J.: ***Supreme Power***. Marvel Comics, 2005
13) Lane G., Guillaumat F.: ***La loi de Bitur-Camember***. 2002
14) Miller F.: Sin City: ***That Yellow Bastard Bk. 4***. Dark Horse Comics, 2005
15) Skála F.: ***Velké putování vlase a brady.*** Arbor Vitae, 2007

Zdroje ilustrací online

16) http://www.applegeeks.com/comics/
17) http://www.dorkboycomics.com/
18) http://jamesblond.transplantcomics.com/
19) http://www.jentong.com/comics.html
20) http://suicidegirls.com/

Zdroje online

21) Wikipedia, the free encyclopedia: ***Comics***.
http://en.wikipedia.org/wiki/Comics [duben, 2007]
22) Willmon J.: ***Commix***. online aplikace, verze z roku 2006. http://www.com-mix.org
23) Thirdframestudios: ***Strip generator***. online aplikace, verze 1.0.2. http://stripgenerator.com
24) Microsoft: ***Avoiding Automatic Scaling***.
http://msdn2.microsoft.com/en-us/library/1bttkazd.aspx [březen, 2007]
25) Cardinal J.: ***Unicode compliant multilingual word breaker***.
http://www.codeproject.com/csharp/breaker.asp [duben, 2007]
26) Assayag I.: ***An API for Google image search***.
http://www.codeproject.com/cs/library/google_image_search_api.asp [únor, 2007]
27) Doherty M.: ***DotNet tips***.
http://homepage.ntlworld.com/mdaudi100/alternate/controls.html [březen, 2007]
28) Powell R. W.: ***BobPowell.net***. http://www.bobpowell.net/bestfit.htm [březen, 2007]
29) Deurbrouck J.: ***Accessing the Photoshop CS Interface via COM***.
http://www.pcpix.com/Photoshop/ [duben, 2007]
30) Wikipedia, the free encyclopedia: ***HSV color space***.
http://en.wikipedia.org/wiki/HSV_color_space [říjen, 2006]
31) Lau D. L.: ***Digital halftoning***. zdrojový kód, verze z roku 9. 6. 1998.
http://www.engr.uky.edu/~dllau/Halftone/HtmlFiles/Software/HTMLfiles/cdod.html
32) Microsoft: ***Developing DPI-Aware Applications***.
http://msdn2.microsoft.com/en-us/library/ms838191.aspx [květen, 2007]

9 Seznam použitých zkratek a symbolů

.NET Framework – programovací prostředí a komponenta pro operační systém ***Windows***; spravuje běh programů, vytvořených pomocí technologie ***Microsoft .NET Framework***.

Bmp – bitmap. Grafický souborový formát.

DPI – dots per inch. Jednotka rozlišení tisku.

C# – jeden z programovacích jazyků, používaných při programování v programovacím prostředí ***.NET Framework***; syntaxe vychází z jazyka ***C++***.

Flash – programovací prostředí pro tvorbu dynamických stránek.

GDI – graphics device interface. Aplikační rozhraní pro programování aplikací, využívajících grafické operace.

Jpeg – Joint photographic experts group. Metoda komprese pro digitální fotografii; zároveň se tak nazývá grafický souborový formát.

HSV – barevný model, ve kterém se barva určuje pomocí složek barevného odstínu ***H***, saturace ***S*** a jasu ***V***.

HTML – hypertext markup language. Značkovací jazyk pro tvorbu internetových stránek.

HTTP – hypertext transfer protocol. Protokol pro přenos informací na internetu.

MSDN – Microsoft developer network. Součást firmy ***Microsoft***; stará se o vztahy s vývojáři software například díky internetovým stránkám ***MSDN library***, které poskytují množství informací o programování v některém z programovacích prostředí této firmy.

PHP – hypertext preprocessor. Programovací jazyk pro tvorbu dynamických stránek.

Png – portable network graphics. Grafický souborový formát.

RGB – barevný model, ve kterém se barva určuje pomocí složek intenzity červené barvy ***R***, zelené barvy ***G*** a modré barvy ***B***. Někdy se v implementaci tohoto barevného modelu dále přidává čtvrtá složka ***Alpha***, která označuje průhlednost.

10 Seznam příloh

Příloha 1: Ukázky vytvořených komiksů

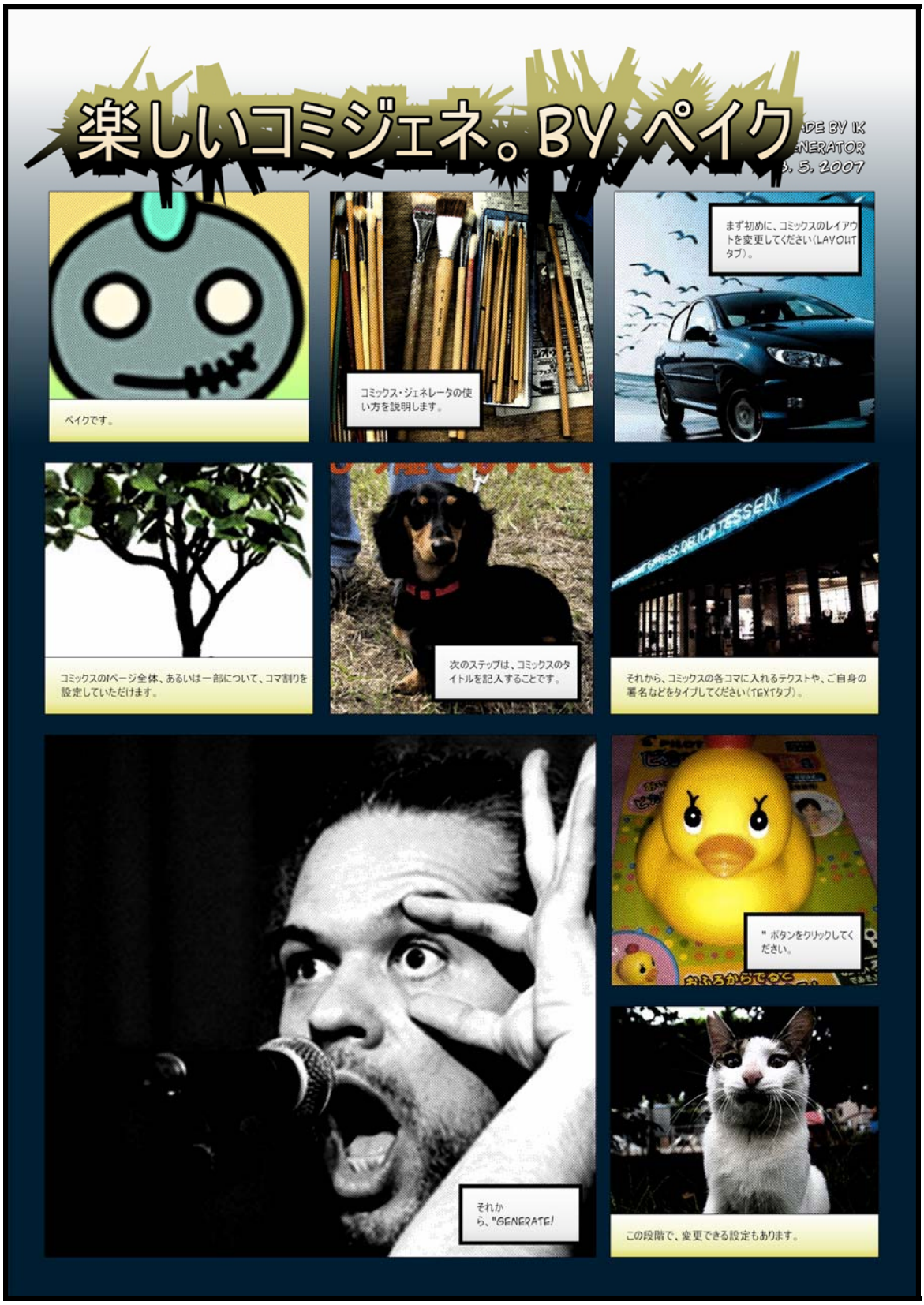
Příloha 2: Postup konfigurace serveru

Příloha 3: CD/DVD

Příloha 1: Ukázky vytvořených komiksů

(Obrázek 37) ukázka komiksu vytvořeného z textu v českém jazyce

(Obrázek 38) ukázka komiksu vytvořeného z textu v anglickém jazyce



(Obrázek 39) ukázka komiksu vytvořeného z textu v japonském jazyce

Příloha 2: Postup konfigurace serveru

Pro úspěšný běh aplikace v serverovém režimu je třeba následujícího postupu:

- V pracovním adresáři aplikace na zvoleném serveru musí být přítomný projektový soubor ***netsettings_default.cmx***, který slouží jako šablona stylu komiksu. Dá se vytvořit tak, že správce serveru vytvoří v naší aplikaci požadovaný styl komiksu a uloží jeho projektový soubor pod tímto názvem.
- V pracovním adresáři mohou být přítomné libovolné ***TrueType*** fonty, které budou aplikací automaticky načteny.
- Klientská aplikace (programovaná v ***PHP***, ***Flash*** nebo jiném jazyce) zjistí některé uživatelem nastavitelné hodnoty a uloží je do souboru ***netsettings.cmx*** ve stejném formátu, v jakém je projektový soubor. V něm jsou jednotlivá nastavení určena textovým řádkem, který obsahuje název proměnné a její hodnotou, oddělené dvojtečkou. Načtení tohoto projektového souboru v aplikaci přepíše nastavení získaná otevřením projektového souboru šablony. Je tedy možné pomocí klientské aplikace nastavit libovolnou podmnožinu všech uživatelem nastavitelných hodnot.
- Klientská aplikace se dále musí postarat o vhodný způsob spuštění aplikace na serveru s parametrem „***-server nazev_obrazku.jpg***“. Je vhodné použít jedinečný název obrázku (například dle aktuálního času), aby nedošlo k přepsání staršího obrázku.
- Klientská aplikace může využít informace ze souboru ***netstatus.log***, ve kterém aplikace ukládá aktuální stav generování vyjádřený v procentech. Po skončení obsahuje řetězec „***Idle***“.
- Klientská aplikace zobrazí vygenerovaný obrázek.

Příklad zápisu do projektového souboru a spuštění aplikace v jazyce PHP:

```
$fileHandle = fopen('netsettings.cmx', 'w') or die("can't open file");
fwrite($fileHandle,"random seed : " . rand()%100000 . "\n");
fwrite($fileHandle,"titleText   : " . $title . "\n");
fwrite($fileHandle,"creditsText : " . $credits . "\n");
fwrite($fileHandle,"textText    : " . str_replace("\r\n","|",$text) . "\n");
fclose($fileHandle);
exec("comics.exe -server img.jpg");
```

Proměnné ***$title***, ***$credits*** a ***$text*** jsou v tomto příkladu získány pomocí předchozího odeslání ***HTML*** formuláře pomocí metody ***POST*** s příslušnými názvy proměnných.