Grafické formáty PCX a TGA

Graphics formats PCX and TGA

David Sůkal

Bakalářská práce
2009

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**David SŮKAL**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Grafické formáty PCX a TGA**

Zásady pro vypracování:

1. **Vytvořte literární rešerši na zadané téma. Ta bude obsahovat historii a vývoj rastrových grafických formátů. Zaměřte se především na formáty PCX a TGA.**
2. **Podrobně popište grafické formáty PCX a TGA. Zaměřte se zejména na jejich strukturu.**
3. **Ze získaných informací vytvořte prezentaci v Powerpointu, která by se dala použít při výuce.**
4. **Proveďte návrh programu, který by dokázal pracovat se soubory PCX a TGA v programovacím jazyce C/C++. Jde zejména o jejich načítání, ukládání a zobrazování.**
5. **Vytvořte program podle předchozího bodu zadání. Vytvořte dokumentaci k programu a zdrojové kódy doplňte komentáři, usnadňující pochopení tohoto kódu.**
6. **Koncepci programu navrhněte tak, aby se se načítání/ukládání formátů PCX a TGA dalo použít i v jiných programech.**

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **ŽÁRA, Jiří, BENEŠ, Bedřich, FELKEL, Petr. Moderní počítačová grafika. 1. vyd. Praha : Computer Press, 1998. 448 s. ISBN 80-7226-049-9.**
2. **LIBERTY, Jesse. Naučte se C++ za 21 dní. Brno : Computer Press, 2007. 796 s. ISBN 978-80-251-1583-1.**
3. **MURRAY, James D., VANRYPER, William. Encyklopedie grafických formátů. 1. vyd. Praha : Computer Press, 1997. 922 s. ISBN 80-7226-033-2.**
4. **Wikipedia contributors. PCX [Internet]. Wikipedia, The Free Encyclopedia; 2008 Dec 12, 23:04 UTC [cited 2009 Jan 16]. Available from: http://en.wikipedia.org/w/index.php?title=PCX&oldid=257586417.**
5. **Wikipedia contributors. Truevision TGA [Internet]. Wikipedia, The Free Encyclopedia; 2009 Jan 10, 07:32 UTC [cited 2009 Jan 16]. Available from: http://en.wikipedia.org/w/index.php?title=Truevision_TGA&oldid=263137981.**

Vedoucí bakalářské práce: **Ing. Pavel Pokorný, Ph.D.**
Ústav aplikované informatiky

Datum zadání bakalářské práce: **20. února 2009**

Termín odevzdání bakalářské práce: **1. června 2009**

Ve Zlíně dne 13. února 2009

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této bakalářské práce je vytvořit aplikaci v programovacím jazyce C/C++, která bude umět pracovat s grafickými formáty typu PCX a TGA (jde především o jejich načítání, ukládání a zobrazování), společně s prezentací těchto formátů pro potřeby výuky počítačové grafiky. Dále je důležité vytvořit k tomuto programu patřičnou dokumentaci a zdrojové kódy doplnit o komentáře, usnadňující pochopení celé aplikace. Teoretická část se zabývá historií a vývojem rastrových formátů, přičemž důraz je kladen na formáty PCX a TGA, u kterých je navíc podrobně zpracován popis jejich struktury.

Klíčová slova: PCX, TGA, C/C++, Visual Studio

ABSTRACT

The aim of the bachelor's thesis is to create application in programming language C/C++, which will deal with graphics formats – type PCX and TGA (their retrieving, saving and displeying) in conjunction with the presentation of these formats for education of computer graphic. Then is important to create for this programme an appropriate documentation and in source codes complete the commentary, which help to understand the whole application. The theoretical part of thesis is engaged in the history and the progress of the raster graphis formats, first of all graphics formats PCX and TGA, which structure is detaily described.

Keywords: PCX, TGA, C/C++, Visual Studio

Poděkování:

Rád bych poděkoval vedoucímu práce Ing. Pavlu Pokornému, Ph.D. za rady a připomínky v průběhu řešení a také svým rodičům a přátelům za trpělivost a podporu při studiu.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

Tato práce se zabývá grafickými formáty PCX a TGA.

Grafické formáty stanovují pravidla, podle kterých je obrázek uložen v souboru. Těchto formátů existuje nepřeberné množství, z nichž každý má své výhody a nevýhody. Takřka každý jednotlivý formát měl svoje opodstatnění a logické místo v počítačové historii.

Toto téma jsem si vybral proto, abych se zdokonalil v programovacím jazyce C/C++ a také proto, abych se pokusil vytvořit poněkud komplexnější a složitější aplikaci.

Práce je rozdělena na teoretickou a praktickou část. V teoretické části jsem se zabýval stručnou historií a popisem bitmapových grafických formátů a zejména detailním popisem formátů PCX a TGA. Shrnul jsem výhody a nevýhody těchto dvou formátů a podrobně popsal jejich strukturu.

V praktické části jsem napsal knihovnu obsahující funkce pro práci s formáty PCX a TGA. Pomocí této knihovny lze obrázky načítat, ukládat a provádět jednoduché operace jako např. inverzi barev nebo převrácení obrazu. Navrhl jsem a vytvořil jednoduchý program využívající tuto knihovnu.

I. TEORETICKÁ ČÁST

1 BITMAPOVÁ GRAFIKA

Bitmapová grafika (někdy také bitmapa či rastrová grafika) je spolu s vektorovou grafikou jedním ze dvou hlavních způsobů, jak lze zaznamenat dvojrozměrný obraz. Celý bitmapový obrázek je tvořen pravidelnou mřížkou z bodů, přičemž každý bod má přiřazenu určitou barvu. Na obrazovce pak jednotlivé barevné body splývají vlivem nedokonalosti lidského oka a uživatel tak vidí pouze barevné plochy, přechody apod. [2]

Obr. 1 - Ukázka bitmapové grafiky

1.1 Definice obrazu v bitmapové grafice

Bitmapový obraz je tvořen sítí jednotlivých bodů - pixelů. Každý pixel má přiřazenu svoji barvu. Důležitým parametrem obrazu je barevná hloubka (určuje počet bitů, kterými je barva bodu popsána). Nejmenší barevnou hloubku má černobílá grafika, kde pro vyjádření stavu bílá a černá stačí každému pixelu pouze jeden bit. U obrázků v obrazovém prostoru RGB má každý pixel alespoň tři byty – pro každou ze základních barev (R – červená, G – zelená, B – modrá) je definována její intenzita. Každá bitmapa musí mít definovánu svou výšku (počet pixelů vertikálně), šířku (počet pixelů horizontálně) a barevnou hloubku. [8]

1.2 Výhody a nevýhody bitmapové grafiky

Je zřejmé, že bitmapová grafika je poměrně náročná na paměť. Z tohoto důvodu se používají různé kompresní formáty, které umožňují datovou velikost obrázku zmenšit. K nejčastějším kompresním formátům pro přenos bitmapové grafiky patří JPG, GIF a PNG. Všechny tři se běžně používají na internetu.

Další nevýhodou bitmapové grafiky je nemožnost měnit velikost obrázku, aniž by tím došlo ke zhoršení jeho kvality. Při větších zvětšeních navíc začíná být patrná bitmapová mřížka (rastr). Tomuto lze částečně zabránit při použití kvalitního zvětšovacího algoritmu. [9]

Obr. 2 - Zhoršení kvality při zvětšení

Velkou výhodou těchto formátů je jejich široká podpora a použitelnost. Pořízení rastrového obrazu je velmi snadné například pomocí digitálního fotoaparátu nebo skeneru. Bitmapové fotografie a ilustrace jdou velice jednoduše upravovat a dočišťovat.

1.3 Využití bitmapové grafiky

Bitmapová grafika se používá tam, kde by vektorová byla příliš komplexní a těžko použitelná. Jedná se o především fotografie a složité obrázky. Využití sahá od drobných grafických prvků na internetových stránkách, přes bitmapové textury aplikované na 3D

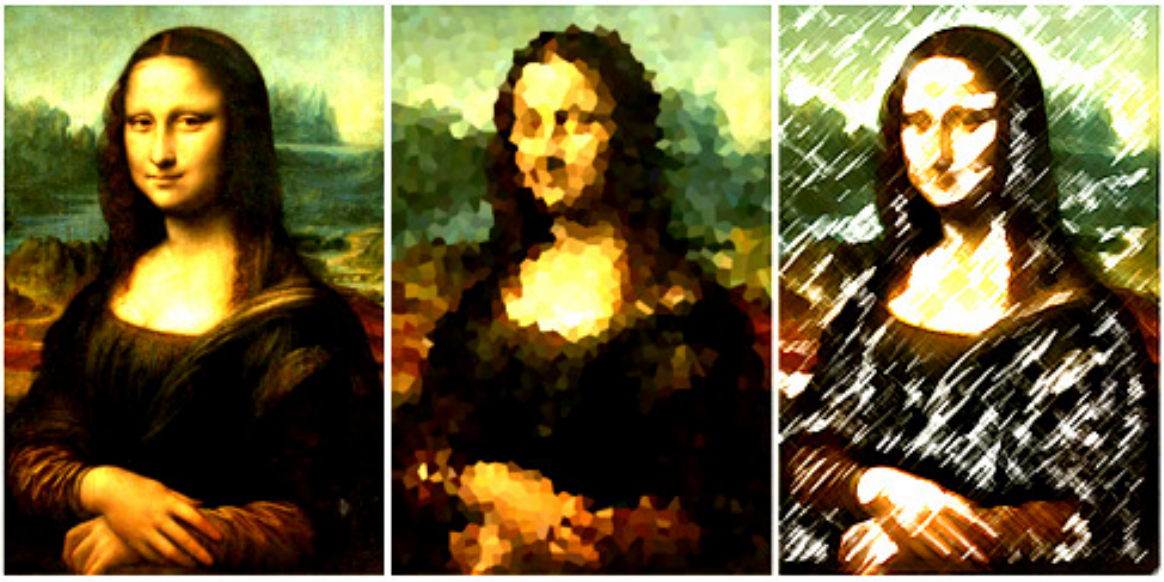
objekty, až po fotografie připravené pro DTP. Na webových stránkách je nejrozšířenějším rastrovým formátem GIF, JPG a PNG. [7]

1.4 Bitmapové editory

Bitmapové editory jsou programy, které slouží pro práci s bitmapovou grafikou pomocí grafického uživatelského rozhraní. Mezi nejznámější patří komerční Adobe Photoshop a volně stažitelný Gimp.

Editory umožňují řadu základních funkcí, mezi něž patří práce v několika vrstvách, práce s textem, několik barevných modelů (způsob mísení základních barev do výsledné barvy – nejpoužívanější jsou RGB a CMYK), vykreslování přímek a základních tvarů, možnost pracovat s mnoha různými grafickými formáty atd.

Komplexnější editory obsahují obrazové filtry pro nejrůznější efekty. Lze například přidávat odrazy a odlesky, simulovat starý vzhled snímků včetně zrna a artefaktů, rozostřovat části či celou fotografii a mnoho dalšího. [8]



Obr. 3 - Ukázka obrazových filtrů

1.5 Bitmapové formáty

Používané formáty souborů rozlišujeme jako nekomprimované a komprimované, komprimované pak na formáty s bezeztrátovou či ztrátovou kompresí. K nejznámějším

formátům patří APNG, BMP, GIF, HDP, JPEG, JPEG, MNG, PCX, PNG, TIFF, WBMP a XPM. [10]

JPEG byl vyvinut skupinou Joint Photographic Experts Group a uznán jako mezinárodní standart v roce 1988. Členy skupiny jsou zástupci řady významných akademických pracovišť a komerčních subjektů (Adobe, Canon, Ericsson, Kodak, Ricoh, Samsung a další). JPEG používá ztrátovou kompresi, která sice snižuje kvalitu obrázku, ale zároveň také snižuje výslednou velikost souboru, což je pro použití na internetu velmi důležité. Ztráty na kvalitě mohou být lidským okem téměř nerozpoznatelné. Poměr komprese vůči kvalitě (a tím i výsledná velikost souboru) se dá ručně nastavit při ukládání. Je vhodný pro ukládání fotografií a obrázků s velkou barevnou hloubkou. Neumožňuje transparentní barvy a nepodporuje animace.

Formát **GIF** (Graphics Interchange Format) byl vyvinut americkou společností CompuServe v roce 1987. Používá bezztrátovou komprimační metodu pojmenovanou jako LZW - Lempel-Ziv-Welch. Patent na tuto kompresi vlastní společnost UNISYS. Výsledná velikost grafického souboru nejvíce závisí na počtu barev, ve kterých se obrázek ukládá. K nejdůležitějším vlastnostem GIFu patří možnost volby jedné barvy jako barvy transparentní a dále možnost vytvářet animace (pohyblivé obrázky). Největší uplatnění nachází při tvorbě grafiky pro použití na internetu [12]

BMP (Windows BitMaP) ukládá souborová data ve formátu Device-Independent Bitmap (DIB). Je nezávislý na provozovaném zařízení a jeho název je odvozen od toho, že jeho barvy jsou popsány ve formátu nezávisle na koncovém výstupním zařízení. Formát nepoužívá žádnou kompresi a je tedy vhodný pro snímky, které mají být v maximální dosažené kvalitě. To vše je na úkor objemu dat.

PNG (Portable Network Graphics) byl vyvinut konsorciem společností na základě bezztrátové komprimační technologie pod názvem deflation. Zvládne uložit až 16-bitovou hloubku stupňů šedi a 48-bitovou hloubku barev (což je 16 bitů pro každou z hodnot RGB na pixel). Umožňuje 16-bitový alfa kanál, 256-stupňové nastavení transparentnosti, ukládá textové informace a informace o křivce "Gamma". PNG používá bezztrátovou kompresi, jejíž schéma neodstraňuje žádné informace o obraze, avšak redukuje množství barev.

TIFF (Tagged Image File Format) je specifický formát pro zachování vysoké kvality obrazu, používaný pro ukládání digitálních snímků bez použití jakékoliv komprese. [12]

2 HISTORIE A VÝVOJ GRAFICKÝCH FORMÁTŮ PCX A TGA

2.1 Historie formátu PCX

Grafický formát PCX vyvinula firma ZSoft Corporation, která ho v minulosti používala ve všech verzích grafického editoru PC-PaintBrush. Tyto aplikace zpočátku pracovaly pouze pod operačním systémem DOS (MS-DOS, DR-DOS apod.), na počítačích kompatibilních s IBM PC a posléze i v Microsoft Windows 3.0 a Microsoft Windows 3.1. Kromě grafických editorů PC-PaintBrush je však tento typ souborů použitý i v mnoha dalších graficky orientovaných aplikacích včetně her.

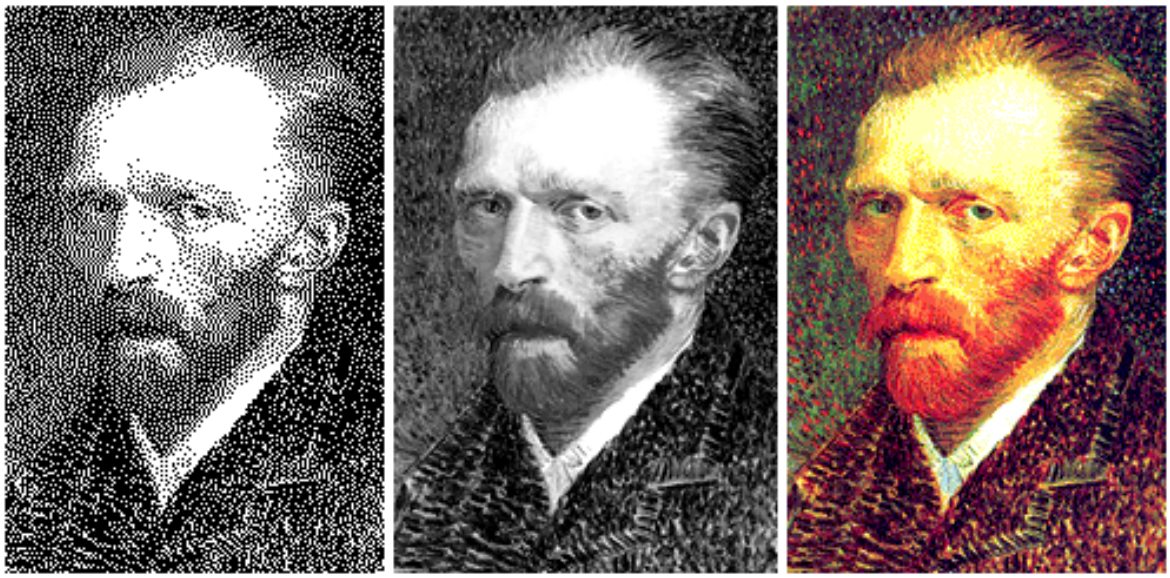
Na tomto grafickém formátu je jasně patrná nekoncepčnost návrhu, samotný formát (hlavička souboru i uložení rastrových dat) totiž kopíroval možnosti tehdejších grafických karet, tj. zejména značně omezený počet barev, organizaci dat do bitových rovin, krátkou barevnou paletu apod. S dalším rozvojem grafických schopností počítačů PC docházelo k nárůstu složitosti celého formátu PCX a různým ad-hoc řešením, například umístěním větší barvové palety nikoli přímo do hlavičky obrázku, ale na samotný konec souboru. [3]

Z těchto důvodů bychom mohli grafický formát PCX považovat spíše za odstrašující příklad. Situace je však složitější, protože se ve své době jednalo o dobře popsaný formát, který navíc nabízel i jednoduše použitelnou komprimaci, která byla při vhodně zvolené barvové paletě pro některé typy obrázků poměrně účinná. Důležitým faktorem je také to, že PC-PaintBrush patřil na PC mezi nejpoužívanější grafické editory.

Z výše uvedených důvodů se PCX stal velmi populární a postupně přešel z počítačů řady PC i na další platformy. Na platformách odlišných od PC byl PCX spíše trpěným formátem. Moderní prohlížeče si s grafickým formátem PCX, který existuje v několika variantách, poradí docela dobře, stejně tak i některé grafické editory. [18]

2.1.1 Základní charakteristika

Formát PCX podporuje barevné režimy bitová mapa, stupně šedi, indexovaná barva a RGB. Režim bitová mapa redukuje obraz na dvě barvy, čímž se zmenší výsledná velikost souboru. Obrázek v režimu indexovaných barev nemá barvy přiřazované přímo, ale jednotlivým pixelům jsou přiřazovány indexy. Každý indexovaný obrázek má svou vlastní mapu barev. RGB je klasický režim, kdy je každému pixelu přiřazena kombinace tří základních barev, která určuje výslednou barvu bodu. Formát PCX nepodporuje alfa kanály. Obrazy mohou mít bitovou hloubku 1, 4, 8 nebo 24 bity.



Obr. 4 - Ukázka barevných režimů

PCX podporuje kompresní metodu RLE (Run Length Encoding). Její princip spočívá v potlačení znaků, které dosáhnou potřebného opakování. Tato metoda je výhodná hlavně v případech, kdy dochází k malým změnám, tj. pro komprimaci obrázků s malou barevnou hloubkou. [13]

2.2 Historie formátu TGA

Grafický formát TGA (Targa) byl navržen firmou Truevision, která několik variant tohoto formátu v minulosti využívala pro ukládání snímků získávaných pomocí svých videograbberů nazvaných Targa. Videograbber je zařízení pro zachytávání a digitalizaci videa, podobnou funkci nabízí dnešní televizní karty. Grabberů typu Targa existovalo několik typů a pro každý typ byla vytvořena varianta vlastní grafického formátu TGA (lišily se především v počtu bitů na pixel).

Později, zejména se stále rostoucí oblibou formátu TGA mezi programátory i uživateli, došlo k dalšímu rozšíření variant způsobů ukládání pixelů a současně i k unifikaci, přičemž výsledkem je dnešní stav, kdy je možné formát TGA použít jak pro rastrové obrázky s barevnou paletou (palette-based images), tak i pro obrázky uložené ve stupních šedi (grayscale) či obrázky typu true color. Podporován je i plnohodnotný osmibitový alfa kanál, pro některé aplikace si však vystačíme s jednobitovým alfa kanálem, resp. maskou průhlednosti.

Díky své jednoduchosti a širokým možnostem se grafický formát TGA značně rozšířil a byl použit v mnoha aplikacích, zejména těch, které musely pracovat s plnobarevnými obrázky. V minulosti, zejména v době operačního systému DOS na počítačích typu PC, se jednalo například o raytracery, ale i některé skenovací programy. [4]

2.2.1 Základní charakteristika

V grafickém formátu typu TGA je možné ukládat bitmapy různých typů. Pravděpodobně nejpoužívanější je nekomprimovaná bitmapa uložená ve skutečných barvách (true color), je však možné uložit i bitmapu ve stupních šedi či bitmapu obsahující místo přímých barev indexy do barvové palety.

Kromě vlastních barev jednotlivých pixelů je možné ukládat i alfa kanál. V něm může být průhlednost popsána buď jedním bitem (podobně jako v případě grafického formátu GIF), nebo bity osmi (256 stupňů průhlednosti). [4]

Rastrový obraz v grafických je v souborech typu TGA uložen buď v komprimované, nebo ve volné (nekomprimované) formě. Je možné komprimovat několika způsoby, typicky se používá jednoduché kódování RLE (Run Length Encoding), které může být kombinované s

Huffmanovým kódem. Komprimované obrazové soubory typu TGA se dnes již téměř nepoužívají a mnoho převodních či prohlížecích programů s nimi ani nedokáže korektně pracovat, resp. podporují pouze variantu RLE a nikoli již Huffmanovo kódování.

TGA je často používán především pro tvorbu fotorealistické grafiky. Většina 3D modelovacích programů tento formát podporuje, a to jak pro výstup finálních obrazů (ze kterých se posléze tvoří video), tak i jako formát vhodný pro uložení textur, protože umožňuje spolu s barevnou informací ukládat i alfa složku (průhlednost), a to dokonce i v obrázcích s barevnou paletou. [19]

Obr. 5 - Ukázka průhlednosti

3 ANATOMIE GRAFICKÉHO FORMÁTU PCX

V následujícím textu je mnohokrát použit výraz little-endian. O co se jedná?

Pojem endianita označuje způsob uložení čísel v paměti počítače, který definuje, v jakém pořadí se uloží jednotlivé byty příslušného datového typu. Označuje se také jako pořadí bytů.

Je jedním z největších zdrojů nekompatibility při ukládání a výměně dat v digitální podobě. Je nutné brát ji v úvahu při přenášení binárních souborů nebo při síťové komunikaci mezi různými platformami. Tento problém pramení z toho, že stejný zdrojový kód zkompilovaný pro počítače s různými procesory může díky jejich různému pořadí bitů produkovat při ukládání nebo přenosu různá binární data. Nejrozšířenějším kódováním vícebytových dat je v současnosti little-endian, což je dáno masovým rozšířením architektury Intel x86.

V případě little-endian se na paměťové místo s nejnižší adresou uloží nejméně významný byt a za něj se ukládají ostatní byty až po nejvíce významný byt. [14]

3.1 Hlavička souboru typu PCX

Grafický soubor typu PCX obsahuje hlavičku, která má délku 128 bytů. Využito bývá pouze 70 bytů (a to ještě ne vždy, zejména nižší verze neobsahují všechny níže popsané informace), zbývajících 58 bytů může využít aplikace, ovšem s tím, že se jejich obsah může kdykoliv, například po konverzi nebo editaci přepsat. Hlavička je tedy využita pouze minimálně, dokonce ještě méně úsporně, než je tomu v grafickém formátu BMP. [3]

Offset	Velikost (byte)	Položka	Význam
0	1	Manufacturer	Vždy obsahuje číslo 0x0a
1	1	Version	Číslo verze
2	1	Encoding	Typ kódování: 0-žádné, 1-RLE
3	1	Bits per pixel	Počet bitů na pixel v obrazové rovině
4	8	Window dimension	Souřadnice obrazu X_1, Y_1, X_2, Y_2 (little-endian)
12	2	Horizontal resolution	Horizontální rozlišení
14	2	Vertical resolution	Vertikální rozlišení
16	48	Color map	Barevná paleta o rozsahu max. 16 barev
64	1	Reserved	Rezervovaná položka
65	1	Color planes	Počet obrazových (bitových) rovin
66	2	Bytes per line	Počet bytů na obrazový řádek
68	2	Palette info	Způsob interpretace palety: 0-barvy, 1-odstíny šedi
70	58	Nonused	Nepoužito, možné využití aplikací
127	...	...	Rastrová data

Tabulka č. 1 - Hlavička souboru typu PCX

První byte hlavičky vždy obsahuje číslo 0x0a. Následuje číslo verze (toto číslo většinou můžeme ignorovat, protože vše potřebné je uloženo v dalších položkách hlavičky). Následuje byte, který určuje zda je použita RLE komprimace, či zda jsou uložena nekomprimovaná rastrová data. Položka Bits per pixel je pojmenovaná poněkud nešťastně, protože neudává celkový počet bitů na pixel, ale počet bitů na pixel v jedné obrazové (bitové) rovině.

Po těchto údajích následují informace o velikosti obrázku a rozlišení. Prvních osm bytů obsahuje krajní souřadnice obrazu X_1, Y_1, X_2 a Y_2. Každá souřadnice zabírá dva byty uložené za sebou v systému little-endian, jak je tomu na procesorech Intel zvykem. Další dva byty obsahují horizontální rozlišení (například pro starší grafické karty je zde hodnota 640 pixelů) následované vertikálním rozlišením (typicky 350, 400, 480 pixelů). Modernější aplikace zde spíš ukládají rozlišení zadané v DPI, například 300 – jde však o jiný význam slova "rozlišení". [3]

Vidíme, že v hlavičce se našlo místo i pro barevnou paletu, ovšem pouze pro šestnáct barev (16×3 = 48 bytů). To postačuje pro ukládání obrázků použitelných na grafických kartách EGA a VGA, ovšem pro obrázky s 256 barvami už kapacita barvové palety nedostačuje.

Po jednom byte, který je rezervován (nemá žádný zvláštní význam a je vhodné ho nulovat), následuje velmi důležitá informace – počet obrazových (bitových) rovin následovaný počtem bytů na obrazový řádek. Počet bitových rovin je velmi důležitý údaj, protože spolu s počtem bitů na pixel udává skutečnou bitovou hloubku obrázku. Za touto informací už následuje pouze jeden významný byte, ve kterém je uloženo, zda se má barevná paleta chápat jako monochromatická nebo barevná (tento byte je však přítomen pouze u novějších verzí PCX). [3]

3.2 Komprimace RLE algoritmem

Rastrová data mohou být v grafickém formátu PCX uložena buď v přímé (tj. nekomprimované) podobě nebo v komprimovaném tvaru. Nekomprimované PCX se prakticky nepoužívají, většina obrázků uložených v tomto formátu používá jedinou podporovanou komprimační metodu – modifikovaný algoritmus RLE (Run Length Encoding).

Zajímavou a velice sympatickou vlastností je to, že algoritmus RLE má u PCX stále stejnou podobu, a to bez ohledu na typ komprimovaného obrázku. Nemusíme tedy rozlišovat, zda se jedná o černo-bílý obrázek, šestnáctibarevný obrázek, obrázek s 256 barvami či plnobarevný (truecolor) obrázek. [5]

RLE použitý u PCX pracuje s proudem bytů, přičemž maximální délka tohoto proudu odpovídá délce obrazového řádku (tato hodnota je uložena v hlavičce). Obrazový řádek se postupně načítá a zjišťuje se, kolik bytů (nikoli pixelů) má stejnou hodnotu. Blok za sebou jdoucích bytů se stejnou hodnotou se zapíše jako dvojice bytů: první byte udává počet znaků v bloku, druhý byte hodnotu těchto bytů. Počitadlo bytů je inicializováno na hodnotu 0xc0, což znamená, že pokud dekomprimační program narazí na byte větší než 0xc0, ví, že se jedná o dvoubytový komprimovaný blok.

Jednotlivé byty, které nejsou součástí bloku a mají hodnotu menší než 0xc0, jsou do komprimovaného souboru zapsány ve své původní podobě. Horší je to s byty, které mají hodnotu větší než 0xc0. Aby nenastala kolize s počitadlem, musí se tyto byty uložit jako dvojice bytů 0xc1 0x??, tj. jako blok o délce jednoho bytu s barvou pixelu uloženou ve druhém bytu. V tomto případě tedy nenastává komprimace, ale naopak prodloužení výstupního souboru. To vede k zajímavému paradoxu, který se u jiných komprimačních

metod neprojevuje: pouhou úpravou barvové palety je možné měnit komprimační poměr u PCX souborů (ideální je, aby paleta byla setříděna tak, že nejčastěji používané barvy jsou uloženy na začátku, aby se omezil počet bytů s hodnotou větší než 0xc0). [5]

4 ANATOMIE GRAFICKÉHO FORMÁTU TGA

4.1 Interní struktura souborů typu TGA

Všechny informace jsou v souborech typu TGA rozděleny do čtyř sekcí, přičemž pouze první sekce je povinná, ostatní sekce mohou či nemusí být použity. Sekce jsou do značné míry podobné blokům v grafickém formátu GIF nebo chunkům ve formátu PNG, ovšem s tím rozdílem, že nemusí obsahovat identifikační hlavičku – pozice sekcí v souboru je možné zjistit již po načtení informační hlavičky souboru. Význam jednotlivých sekcí je následující:

i. V první sekci umístěné na začátku souboru je uložena informační hlavička, jejíž velikost je vždy rovna 18 bytům. V hlavičce jsou umístěny základní informace o obraze, zejména jeho rozlišení, způsob kódování barev pixelů a orientace obrázku.

ii. Za informační hlavičkou může následovat identifikační pole obrázku, což je textový řetězec o maximální délce 255 znaků. Tato sekce je však nepovinná a málokdy se s ní v obrazových souborech setkáváme.

iii. Ve třetí sekci může být uložena barevná paleta. Tato sekce je, podobně jako sekce předchozí, opět nepovinná. Používá se pouze u některých obrázků s formátem 8 bitů na pixel.

iv. V sekci čtvrté jsou uložena vlastní rastrová data, tj. barvy jednotlivých pixelů. Posloupnost rastrových dat (zejména orientaci vertikální osy) lze ve formátu TGA specifikovat přímo v hlavičce, je například možné obrázky ukládat od prvního řádku do řádku posledního či naopak. Rastrová data mohou být komprimována jednoduchým RLE algoritmem. [4]

4.1.1 Hlavička souboru typu TGA

Na začátku všech souborů typu TGA je uložena informační hlavička, která má vždy velikost 18 bytů. V této hlavičce jsou specifikovány všechny důležité informace o rastrovém obraze a způsobu jeho uložení v souboru.

Offset	Velikost (byte)	Název	Význam
0	1	IDLength	Velikost obrazového identifikátoru
1	1	ColorMapType	Typ barevné palety
2	1	ImageType	Typ obrázku
3	2	CMapStart	Počátek barevné palety
5	2	CMapLength	Délka barevné palety
7	1	CMapDepth	Bitová hloubka položek barevné palety
8	2	XOffset	X-ová souřadnice počátku obrázku
10	2	Yoffset	Y-ová souřadnice počátku obrázku
12	2	Width	Šířka obrázku uvedená v pixelech
14	2	Height	Výška obrázku uvedená v pixelech
16	1	PixelDepth	Bitová hloubka
17	1	ImageDescriptor	Popisovač obrázku

Tabulka č. 2 - Hlavička souboru typu TGA

Význam jednotlivých položek v hlavičce je následující:

Položka IDLength obsahuje počet bytů v identifikačním poli obrázku. Pokud je hodnota této položky nulová, identifikační pole není použito.

Položka ColorMapType může nabývat pouze dvou hodnot. Pokud se nepoužívá barevná paleta, je zde uložena nula (0x00), pokud se barevná paleta používá, je zde uložena jednička (0x01). [4]

Položka ImageType obsahuje informace o formátu uložení a kódování rastrových dat:

Hodnota	Význam
0	Žádná rastrová data nejsou uložena
1	Nekomprimovaná data s barevnou paletou
2	Nekomprimovaná data ve formátu RGB
3	Nekomprimovaná data v odstínech šedi
9	Data kódovaná RLE s barevnou paletou (je nastaven bit 0x08)
10	Data kódovaná RLE ve formátu RGB (je nastaven bit 0x08)
11	Data kódovaná RLE v odstínech šedi (je nastaven bit 0x08)
32	Data kódovaná Huffmanovým kódem a RLE s barevnou paletou
33	Data kódovaná Huffmanovým kódem a RLE s barevnou paletou (uložení v quadtree)

Tabulka č. 3 - Formáty uložení dat v TGA

Položka CMapStart obsahuje index první barvy v barvové paletě.

Položka CMapLength obsahuje počet položek uložených v barvové paletě.

Položka CMapDepth obsahuje počet bitů pro každou položku v paletě. Mohou zde být uloženy hodnoty 0 (bez palety), 16 (15 bitů pro barvu, jeden bit pro alfa kanál), 24 (plné RGB) nebo 32 (RGBA).

V položce XOffset je uložena X-ová souřadnice levého spodního rohu obrázku. Tato hodnota může být použita pro ukládání výřezů i s jejich relativním umístěním v originálním obrázku, i když tuto možnost mnoho programů nepoužívá.

V položce YOffset je uložena Y-ová souřadnice levého spodního rohu obrázku. Význam této hodnoty je stejný jako u předchozí položky.

Položka Width obsahuje šířku obrázku v pixelech. Maximální šířka je tak teoreticky 65535 pixelů, v praxi však některé programy používají 16ti bitová čísla se znaménkem (short int), a tak je vhodné omezit šířku pouze na 32767 pixelů.

Položka Height obsahuje výšku obrázku v pixelech. Maximální výška je opět omezena na 32767 pixelů.

Položka PixelDepth obsahuje počet bitů na jeden pixel. Podle typu obrázku (tj. maximálního počtu barev) zde mohou být hodnoty 1bpp, 8bpp, 16bpp, 24bpp a 32bpp (bpp – bits per pixel).

V položce ImageDescriptor jsou uloženy příznaky (flags) specifikující posloupnost uložených pixelů. [4]

4.1.2 Identifikační pole obrázku

Identifikační pole obrázku je představováno řetězcem, jehož formát je volný, tj. není žádnou normou či specifikací pevně stanoven. Proto může každá aplikace do tohoto pole ukládat libovolné údaje. Maximální délka řetězce je 255 znaků, protože velikost položky IDLength je pouze 1 byte a nulová hodnota (0x00) je rezervována pro případ, že by aplikace identifikační pole obrázku nevyplnila. Většina aplikací identifikační pole nepoužívá, proto do IDLength nastavuje nulovou hodnotu, nicméně při načítání obrázků typu TGA je vhodné hodnotu uloženou v této položce načítat a pole přeskakovat.

Při pohledu na způsob uložení identifikačního pole v souborech typu TGA vidíme, že není vyřešeno k plné spokojenosti, protože se toto pole může kdykoli přepsat a aplikace se tedy nemohou spolehnout na jeho obsah. [4]

4.1.3 Barevná paleta

Barevná paleta obsahuje hodnoty barevných složek RGB pro každou položku uloženou v paletě. Celkový počet položek v paletě je zadán v hlavičce souboru TGA atributem CMapLength, index první položky potom atributem CMapStart. Podle počtu bitů rezervovaných na jednu položku (atribut CMapDepth) jsou povoleny tři formáty položek v barvové paletě:

1. **32 bitů**: barevné složky RGB spolu s alfa-složkou (průhledností) jsou za sebou uloženy v pořadí modrá, zelená, červená a alfa. Každá složka má velikost jeden byte.
2. **24 bitů**: barevné složky RGB jsou za sebou uloženy v pořadí modrá, zelená a červená, tj. ve skutečnosti jde o formát BGR. Každá složka má velikost opět jeden byte.
3. **16 bitů**: barevné složky RGB jsou spolu s příznakem průhlednosti (pouze 1 bit, tj. buď 0% nebo 100% průhlednosti) uloženy v bitové struktuře: $AR_4R_3R_2R_1R_0G_4G_3$ $G_2G_1G_0B_4B_3B_2B_1B_0$. Bity $R_4R_3R_2R_1R_0$ představují červenou barevnou složku, bity $G_4G_3G_2G_1G_0$ zelenou barevnou složku a bity $B_4B_3B_2B_1B_0$ modrou barevnou složku modrou.

4.1.4 Rastrová data

Formát uložení rastrových dat je vždy typu little-endian, tj. v konvencích používaných u procesorů Intel.

Pro nekomprimované obrázky uložené ve formátu 1bpp je každý pixel reprezentován jedním bitem, které se slučují po osmicích do jednoho bytu. Tento formát není příliš podporován.

Pro nekomprimované obrázky uložené ve formátu 8bpp (tj. buď paletové obrázky nebo obrázky ve stupních šedi) je každý pixel v rastru reprezentován jedním bytem.

Pro nekomprimované obrázky uložené ve formátu 16bpp (vždy bez barevné palety) je každý pixel uložen ve dvou bytech, které mají bitovou strukturu $AR_4R_3R_2R_1R_0G_4G_3$

$G_2G_1G_0B_4B_3B_2B_1B_0$, kde A odpovídá bitu průhlednosti, R_x jsou jednotlivé bity červené barvové složky, G_x zelené barvové složky a B_x barvové složky modré.

Nekomprimované obrázky v barevné hloubce 24bpp nebo 32bpp (samozřejmě v tomto případě bez barevné palety) jsou uloženy tak, že každý pixel je představován trojicí bytů RGB, popř. čtveřicí bytů ARGB. Vše je uloženo dle konvence procesorů Intel, tj. BGR a BGRA. [4]

4.2 Typy grafického formátu TGA

4.2.1 1 bpp (black and white)

Nejjednodušším typem grafického souboru typu TGA jsou obrázky nazývané anglickým souslovím black and white. Opravdu se jedná o obrázky obsahující pouze plně černé a plně bílé pixely.

Pixely jsou v obrázcích typu black and white uloženy po osmicích, které tvoří jeden byte. Každý jednotlivý bit z bytu slouží pro uložení barvy jednoho pixelu – černá barva je uložena jako bit s hodnotou nula a barva bílá jako bit s hodnotou jedna, z toho také vychází výše uvedené označení 1 bpp (jeden bit na pixel). U černobílých TGA obrázků je zajímavé, že je některé prohlížecí programy zobrazují inverzně, tj. pixely uložené s nulovou hodnotou jsou zobrazeny bílou barvou a pixely s jednotkovou hodnotou barvou černou.

Některé programy však mají s tímto typem obrázků problémy a vůbec ho nezobrazí. Těžko říci, zda se jedná o akceptovatelné či zcela nevhodné chování, protože zrovna tento typ obrázků firma Truevision opravdu nikdy ve svých grabberech nepoužila, jedná se tedy o jakési ideové rozšíření původních tří typů obrázků Targa, které nebylo touto firmou oficiálně schváleno. [10]

Hodnoty bytů hlavičky	Význam
00	Obrázek je uložen bez identifikačního pole
00	Barevná paleta není přítomna
03	Obrázek je uložen ve stupních šedi (grayscale)
00 00	Počátek barvové palety=0
00 00	Délka barvové palety je nulová (paleta nepoužita)
00	Počet bitů na jednu položku palety je nulový
00 00 00 00	Umístění obrázku do počátku souřadné soustavy
XX XX	Šířka obrázku v pixelech (little-endian)
YY YY	Výška obrázku v pixelech (little-endian)
01	Počet bitů na pixel (bpp) je roven 1
20	Obrázek je uložen shora dolů
...	Rastrová data po řádcích 1 bit na pixel

Tabulka č. 4 - Hlavička TGA s 1 bpp

4.2.2 8 bpp ve stupních šedi (grayscale)

Obrázky uložené ve stupních šedi mají také jednoduchý formát. Barevná paleta není uložena a vlastní rastr pixelů má v případě nepoužití komprimace formát "co byte, to jeden pixel". S těmito typy obrázků se již můžeme v praxi často setkat, protože například mnoho ručních skenerů (levnější a starší varianty skenerů stolních), resp. jejich ovladačů, ukládalo naskenované obrázky právě jako grayscale TGA. Můžeme se setkat se dvěma variantami, podle toho, zda je obrázek uložen shora dolů (úprava pro běžné grafické karty) či naopak (původní formát firmy Truevision). [6]

Hodnoty bytů hlavičky	Význam
00	Obrázek je uložen bez identifikačního pole
00	Barevná paleta není přítomna
03	Obrázek je uložen ve stupních šedi (grayscale)
00 00	Počátek barvové palety=0
00 00	Délka barvové palety je nulová (paleta nepoužita)
00	Počet bitů na jednu položku palety je nulový
00 00 00 00	Umístění obrázku do počátku souřadné soustavy
XX XX	Šířka obrázku v pixelech (little-endian)
YY YY	Výška obrázku v pixelech (little-endian)
08	Počet bitů na pixel (bpp) je roven 8
20	Obrázek je uložen shora dolů
...	Rastrová data po řádcích 1 byte na pixel

Tabulka č. 5 - Hlavička TGA s 8 bpp ve stupních šedi

4.2.3 8 bpp s barevnou paletou

Obrázky TGA s barevnou paletou nejsou tak rozšířené, jak by se na první pohled mohlo zdát. V tomto případě se však nejedná o nějaké nepřekonatelné technické limity TGA, spíše se opět musíme podívat po praktických důvodech. TGA se rozšířilo zejména jako formát vhodný pro práci s plnobarevnými obrázky, tj. 24bpp a později i 32bpp. Při práci s obrázky využívajícími barevnou paletu se však již před nástupem TGA používaly formáty jiné, zejména PCX a GIF.

V každém případě je paletový režim charakterizovaný tím, že pixely uložené v rastru obsahují index do barvové palety, který leží v rozsahu od 0 do 255. Barevná paleta má maximální délku 256 položek, ale může být i menší, například v případě, že je použito pouze 128 barev. Každá položka v barvové paletě má délku tří či čtyř bytů. Pokud je dlouhá tři byty, obsahuje barvové složky RGB, v případě, že má délku čtyř bytů, může být ke každé barvě přiřazena i průhlednost (jedná se o plnohodnotný osmibitový alfa kanál). [6]

Hodnoty bytů hlavičky	Význam
00	Obrázek je uložen bez identifikačního pole
01	Barevná paleta je přítomna
01	Obrázek je uložen v paletovém režimu
00 00	Počátek barvové palety=0
00 01	Délka barvové palety je rovna 256 položkám
18	Počet bitů na jednu položku palety je 24
00 00 00 00	Umístění obrázku do počátku souřadné soustavy
XX XX	Šířka obrázku v pixelech (little-endian)
YY YY	Výška obrázku v pixelech (little-endian)
08	Počet bitů na pixel (bpp) je roven 8
20	Obrázek je uložen shora dolů
...	Rastrová data po řádcích 1 byte na pixel

Tabulka č. 6 - Hlavička TGA s 8 bpp

4.2.4 16 bpp s jednobitovou průhledností (hi-color)

Dnes se nejčastěji můžeme setkat s obrázky typu TGA, které mají pro každý pixel vyhrazeno 16 bitů nebo 24 bitů – ostatně kvůli těmto režimům je TGA oblíben. V prvním případě se jedná o takzvané hi-color obrázky, ve druhém případě o obrázky true color (pravé barvy). Jedná se o bezpaletové obrázky, tj. délka barevné palety je u obou typů nulová. Formát 16bpp je často použit pro ukládání textur, protože menší bitová šířka jednotlivých barvových kanálů (v tomto případě se jedná o pět bitů) pro mnohé textury

vyhovuje a navíc je možné využít jednobitový alfa kanál pro vytvoření "děr" v textuře – příkladem mohou být různá okna nebo mříže.

U obrázků uložených ve formátu 16bpp je každý pixel zapsán ve dvou bytech, které mají bitovou strukturu $AR_4R_3R_2R_1R_0G_4G_3G_2G_1G_0B_4B_3B_2B_1B_0$, kde A odpovídá bitu průhlednosti, R_X jsou jednotlivé bity červené barvové složky, G_X zelené barvové složky a B_X barvové složky modré. Dvojice bytů je uspořádána v pořadí little-endian, což je pro procesory řady x86 přirozené. [6]

Hodnoty bytů hlavičky	**Význam**
00	Obrázek je uložen bez identifikačního pole
00	Barevná paleta není přítomna
02	Obrázek je uložen v true color režimu
00 00	Počátek barvové palety=0
00 00	Délka barvové palety je nulová
00	Počet bitů na jednu položku palety je 0
00 00 00 00	Umístění obrázku do počátku souřadné soustavy
XX XX	Šířka obrázku v pixelech (little-endian)
YY YY	Výška obrázku v pixelech (little-endian)
10	Počet bitů na pixel (bpp) je roven 16
21	Obrázek je uložen shora dolů a počet alfa bitů je 1
...	Rastrová data po řádcích 2 byte na pixel

Tabulka č. 7 - Hlavička TGA s 16 bpp

4.2.5 24 bpp (true color bez alfa kanálu)

Nekomprimované obrázky uložené v barevné hloubce 24bpp jsou strukturovány tak, že každý pixel je představován trojicí bytů tvořících barvu v prostoru RGB (Red, Green, Blue). Vše je uloženo opět dle konvence procesorů Intel, tj. BGR (Blue, Green. Red).

Někdy se také setkáme s plnobarevným obrázkem otočeným "hlavou dolů". Tuto skutečnost je možné zjistit z posledního bytu hlavičky (v souboru je uložen na osmnácté pozici), protože pátý bit ukazuje, zda je obrázek uložen v otočené podobě či v podobě vhodnější pro další zpracování a zobrazení. Pro test, zda se jedná o zrcadlený obrázek, postačuje provést bitový logický součin hodnoty posledního bytu hlavičky konstantou 0x20. [6]

Hodnoty bytů hlavičky	Význam
00	Obrázek je uložen bez identifikačního pole
00	Barevná paleta není přítomna
02	Obrázek je uložen v true color režimu
00 00	Počátek barvové palety=0
00 00	Délka barvové palety je nulová
00	Počet bitů na jednu položku palety je 0
00 00 00 00	Umístění obrázku do počátku souřadné soustavy
XX XX	Šířka obrázku v pixelech (little-endian)
YY YY	Výška obrázku v pixelech (little-endian)
18	Počet bitů na pixel (bpp) je roven 24
20	Obrázek je uložen shora dolů a počet alfa bitů je 1
...	Rastrová data po řádcích 3 byte na pixel

Tabulka č. 8 - Hlavička TGA s 24 bpp

4.2.6 32 bpp (true color s alfa kanálem)

Obdobně jako obrázky RGB (24 bpp) je uložen i obrázek ve formátu RGBA, tj. s 32 bity na pixel. Formát pixelů je v tomto případě BGRA, tj. nejméně významný byt je uložen jako první. V hlavičce nastává změna v předposledním a posledním bytu (sedmnáctá a osmnáctá pozice od počátku souboru). V sedmnáctém bytu hlavičky se počet bitů na pixel nastaví na hodnotu 0x20 (32bpp) a v osmnáctém bytu se specifikuje bitová hloubka alfa kanálu na 8. [6]

4.3 Komprimace v souborech typu TGA

V grafických souborech typu TGA lze ukládat i rastrové obrázky komprimované jednoduchým kódem RLE (Run Length Encoding). Kódování RLE je použito pouze u obrázků, které mají v informační hlavičce nastavenou hodnotu ImageType na 9, 10 nebo 11, u obrázků typu 32 a 33 je použita kombinace RLE s Huffmanovým kódem. Při použití RLE kódování jsou posloupnosti po sobě následujících pixelů s barvami se stejnou hodnotou zakódovány dvojicí hodnot: počtem opakování a hodnotou opakování. Podobný způsob komprimace nabízí i grafické formáty BMP a PCX, konkrétní implementace se však poněkud liší. [6]

V režimu kódování RLE je nejvyšší bit (tj. hodnota 0x80 hexadecimálně) každého bytu na začátku datového paketu rezervován jako logický příznak, zda se bude jednat o opakování (RLE paket) či nikoliv (Raw paket):

1. Pokud je tento příznak nastaven (tj. v byte je uložena hodnota větší než 0x7f), udávají hodnoty nižších sedmi bitů počet opakování (počet opakování=hodnota & 0x7f + 1) hodnoty/barvy uložené v následujících bytech (1 byte pro osmibitové obrázky s paletou i grayscale, 2 byte pro šestnáctibitové obrázky atd.). Opakovaná hodnota není omezena pouze na jeden řádek, může přeskakovat přes více řádků, což je výhodné zejména při ukládání pozadí obrázku (obloha apod.), které je na levém i pravém okraji obrázku vykresleno stejným barevným odstínem. Tato datová posloupnost se nazývá RLE paket.

2. V případě, že je příznak vynulován (tj. v prvním byte posloupnosti je uložena hodnota menší než 0x80), udávají hodnoty nižších sedmi bitů délku dat, které nejsou pomocí RLE kódované. Pokud se tedy v obraze vyskytne složitý motiv, je možné uložit až 128 pixelů různých barev v jednom nekomprimovaném shluku. Tato datová posloupnost se nazývá Raw paket.

U běžných obrázků dává RLE metoda použitá u TGA lepší výsledky než podobné metody použité u grafických formátů BMP a zejména u PCX. Je to z toho důvodu, že složité motivy (po sobě jdoucí pixely s různou barvou) není možné např. v PCX zakódovat jinak než pomocí RLE paketů s jednotkovou délkou. V případě TGA se však až 128 různobarevných pixelů zakóduje do posloupnosti o délce 129 bytů, takže nárůst délky souboru je nepatrný. [6]

5 POUŽITÉ NÁSTROJE PŘI VÝVOJI PROGRAMU

5.1 Visual Studio 2008

Microsoft Visual Studio je vývojové prostředí (IDE) od Microsoftu. Může být použito pro vývoj konzolových aplikací a aplikací s grafickým rozhraním spolu s Windows Forms aplikacemi, webovými stránkami, webovými aplikacemi a webovými službami jak ve strojovém kódu, tak ve spravovaném kódu na platformách Microsoft Windows, Windows Mobile, Windows CE, .NET, .NET Compact Framework a Microsoft Silverlight. [15, 16]

Visual Studio obsahuje editor kódu podporující IntelliSense a refaktorování. Integrovaný debugger pracuje jak na úrovni kódu, tak na úrovni stroje. Další vestavěné nástroje zahrnují designer formulářů pro tvorbu GUI aplikací, designer webu, tříd a databázových schémat.

Visual Studio podporuje jazyky prostřednictvím jazykových služeb, což umožňuje, aby editor kódu a debugger podporoval jakýkoliv programovací jazyk. Mezi vestavěné jazyky patří C/C++ (použitím Visual C++), VB.NET (použitím Visual Basic .NET) a C# (použitím Visual C#). Podpora dalších jazyků jako Chrome, F#, Python a Ruby spolu s ostatními může být přidána jazykovými službami, které musí být nainstalovány zvlášť. Také je podporováno XML/XSLT, HTML/XHTML, JavaScript a CSS. Existují i verze Visual Studia pro určitý jazyk, které uživateli poskytují omezenější jazykové služby. Tyto individuální balíčky jsou Microsoft Visual Basic, Visual J#, Visual C# a Visual C++. [15]

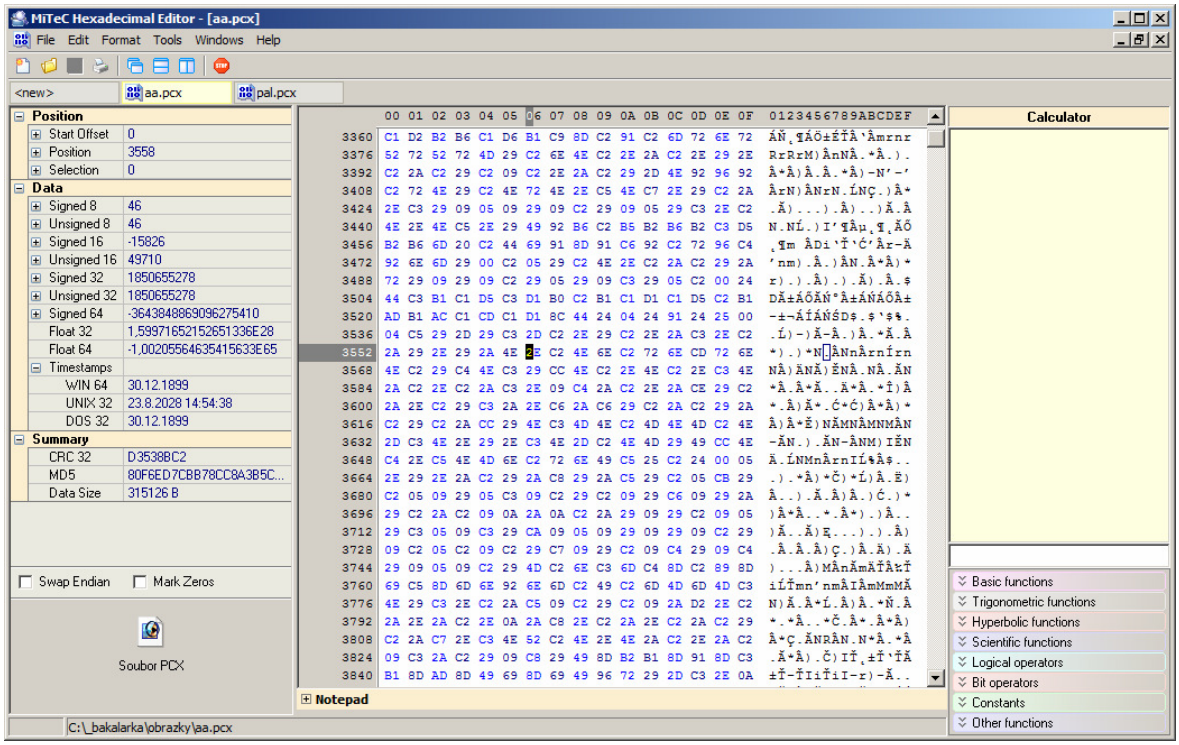
5.1.1 Microsoft .NET Framework

Tento balíček je nutný pro spuštění programů vyvinutých pro prostředí Microsoft .NET. Sám o sobě tento balíček žádnou funkci nemá, tudíž by ani neměl počítač nijak zatěžovat nebo zahlcovat. Aktuálně je tento balíček ve verzi 3.5, ale pro účely programu postačí verze 2.0 z roku 2005 díky tomu, že Visual Studio 2008 dovoluje díky společnému jádru psát programy i pro starší verze Microsoft .NET. Tato funkce se nazývá multitargeting.

5.2 Hexadecimální editor MiTeC

Pro prohlížení souborů v hexadecimálním tvaru jsem zvolil program MiTec, který je volně ke stažení. Obrovskou výhodou tohoto programu je snadnost a přehlednost použití a

v neposlední řadě také možnost současného editování několika souborů. Program není vhodný pro práci s velkými soubory. [17]



Obr. 6 - Hexadecimální editor MiTec

II. PRAKTICKÁ ČÁST

6 KNIHOVNA

Program se skládá ze dvou částí – z knihovny, která obsahuje funkce pro práci se soubory PCX a TGA a z aplikace, která tuto knihovnu využívá. Díky důslednému oddělení těchto dvou částí lze knihovnu použít i v jiných programech. Knihovna obsahuje funkce umožňující načítání/ukládání a také některé základní operace jako je převrácení obrazu nebo inverze barev.

Knihovna obsahuje řadu funkcí pro práci se soubory PCX a TGA. Pomocí funkcí lze bezproblémově načítat a ukládat 24 bitové obrázky typu PCX, načítat plnobarevné, v odstínech šedi a s barevnou paletou typu TGA a také je ukládat. Pro plnobarevnou variantu lze ukládat s použitím RLE komprese nebo bez ní. Kromě těchto funkcí obsahuje knihovna ještě doplňující funkce pro inverzi barev a pro horizontální a vertikální přetočení obrazu.

Použití funkcí je velice jednoduché. Načítání spočívá v převodu příslušného souboru na klasický formát BMP, se kterým může uživatel dále pracovat. Při uložení probíhá převod z tohoto BMP do příslušného formátu.

Nevýhodou výše zmíněného přístupu je nutnost vytvoření reálného souboru BMP na disku, nelze tedy pracovat s pouhým datovým proudem nebo např. dynamickým polem.

Všechny funkce mají dva vstupní argumenty – první je řetězec určující vstupní soubor a druhým argumentem je řetězec pro výstupní soubor. Volání funkce pro převod 24 bitového TGA do 24 bitového BMP vypadá např. následovně:

```
TGA24RLEtoBMP24("C:/vtup.tga", "C:/vystup.bmp");
```

Funkce načte soubor vstup.tga a vytvoří spustitelný soubor vystup.bmp, v případě, že nemáme potřebná práva nebo vstupní soubor neexistuje, je funkce ukončena.

V následující části bude funkce TGA24RLEtoBMP24 rozebrána podrobněji.

6.1 Popis funkce pro načítání 24 bitového TGA

Nejprve je otevřen soubor TGA ke čtení. Pokud nemůže být datový proud vytvořen, je funkce ukončena příkazem return s příslušnou chybovou hláškou. Pokud je otevření souboru úspěšné, je dynamicky vytvořeno pole pabFileData, které svou velikostí odpovídá velikosti TGA. Poté je načten obsah souboru do pole a datový proud pro čtení souboru je uzavřen příkazem fclose.

Poté je vytvořena struktura, která svým složením odpovídá hlavičce TGA. Z proměnné pabFileData je zkopírováno tolik prvních bitů, kolik jich má hlavička. Nyní můžeme k jednotlivým položkám hlavičky jednoduše přistupovat. Například výpis šířky načteného obrázku vypadá následovně:

```
printf("SIRKA: %i",sHlavickaTGA.Width);
```

Poté je vytvořen výstupní soubor, v případě, že nemáme potřebná práva pro zápis a soubor nemůže být vytvořen, je funkce ukončena příkazem return s příslušnou hláškou. Poté je vytvořena struktura, která odpovídá hlavičce obrázku BMP. Parametry jsou k jednotlivým položkám jednoduše přiřazeny.

```
sHlavickaBMP.biHeight = sHlavickaTGA.Height;
```

Po vyplnění všech potřebných parametrů je hlavička zapsána do souboru BMP. Výsledkem je soubor s platnou hlavičkou, který má velikost 54 bytů, což odpovídá délce hlavičky. Nyní je potřeba dodat vlastní obrazová data.

Do souboru BMP se data zapisují po řádcích, kdy každý řádek má stejnou délku. Délka řádku je počet pixelů v řádku (počet pixelů na řádek je v hlavičce TGA) násobený třemi. Výsledné číslo musí být dělitelné čtyřmi, to se provede jednoduchým výpočtem:

```
int iDelkaRadku = sHlavickaTGA.Width*3;
if(iDelkaRadku%4 != 0) iDelkaRadku = iDelkaRadku + (4-iDelkaRadku%4);
```

Když víme, kolika byty je tvořen každý řádek v souboru BMP, dynamicky vytvoříme pole proměnných typu char, které tento řádek reprezentuje v programu.

```
unsigned char *radek = new unsigned char[iDelkaRadku];
```

V další části se program opakuje tolika cykly, kolik má obrázek řádků (tato vlastnost je uložena v hlavičce TGA).

Na začátku každého cyklu je celý řádek (proměnná radek, kterou jsme dynamicky deklarovali výše) vynulován. Dělá se to proto, že délka řádku nemusí přesně odpovídat počtu bytů, které nesou hodnotu o barvě. V případě, kdy není trojnásobek počtu pixelů na řádek dělitelný čtyřmi, zvyšujeme délku řádku BMP. Vznikají pak prázdná místa, která neobsahují žádnou informaci a ta by měla být vynulována.

Dále následuje cyklus, který probíhá tak dlouho, dokud nezaplníme jeden řádek byty, které nesou informaci o barvě. Do pomocné proměnné ukládáme aktuální hodnotu bytu původního obrázku TGA. Proměnná pozice je index na aktuální místo, kde se ve sledu bytů pohybujeme, po každém načtení se jeho hodnota zvyšuje o jedničku.

```
unsigned char hodnota = pabFileData[pozice++];
```

Podle hodnoty pomocné proměnné se pak rozhodujeme dále. Je použita RLE komprese, tudíž je nutno s ní počítat.

V případě, že je hodnota pomocné proměnné větší než 127, následuje sled totožných pixelů. Počet těchto opakování je roven hodnotě rozdílu pomocné proměnné a čísla 127. Trojice bytů, která nese informace o červené, zelené a modré složce pixelu je do řádku zapsána tolikrát, kolik je počet opakování.

V případě, že je hodnota pomocné proměnné menší nebo rovna číslu 127, následuje sled různých pixelů. Počet opakování je roven součtu hodnoty pomocné proměnné a čísla 1. Poté je do řádku zapsáno tolik trojic bytů, které nesou informace o třech základních složkách barvy pixelu, kolik je počet opakování.

Po zaplnění všech bytů v řádku, které nesou informaci, je řádek zapsán do výstupního souboru a cyklus se opakuje pro další řádek.

```
fwrite(radek, sizeof(unsigned char), iDelkaRadku, outFile);
```

Po skončení cyklu je zavřen datový proud pro výstupní soubor a jsou smazána dynamicky alokovaná pole pabFileData a radek.

6.2 Popisy převodních algoritmů

Následuje popis jednotlivých převáděcích algoritmů.

6.2.1 24 bitové TGA do 24 bitového BMP

Jedná se o variantu bez RLE komprese, tudíž je převod velice jednoduchý. Nejprve je načtena hlavička vstupního souboru stejně jako v předchozím případě. Je vytvořena a zapsána hlavička BMP souboru.

Jednoduchým cyklem procházíme obrazová data byt po bytu a vkládáme do proměnné, která nám zastupuje jeden řádek výstupního souboru. Po zaplnění této proměnné ji zapíšeme do výstupního BMP. Pozor si musíme dát pouze na možný rozdíl počtu bytů, které zastupují jeden řádek ve vstupním a výstupním souboru. Počet bytů na řádek může být v BMP vyšší, přebývající místa je nutno vyplnit nulami.

V podstatě se jedná o pouhé překopírování bytů, které nesou informaci.

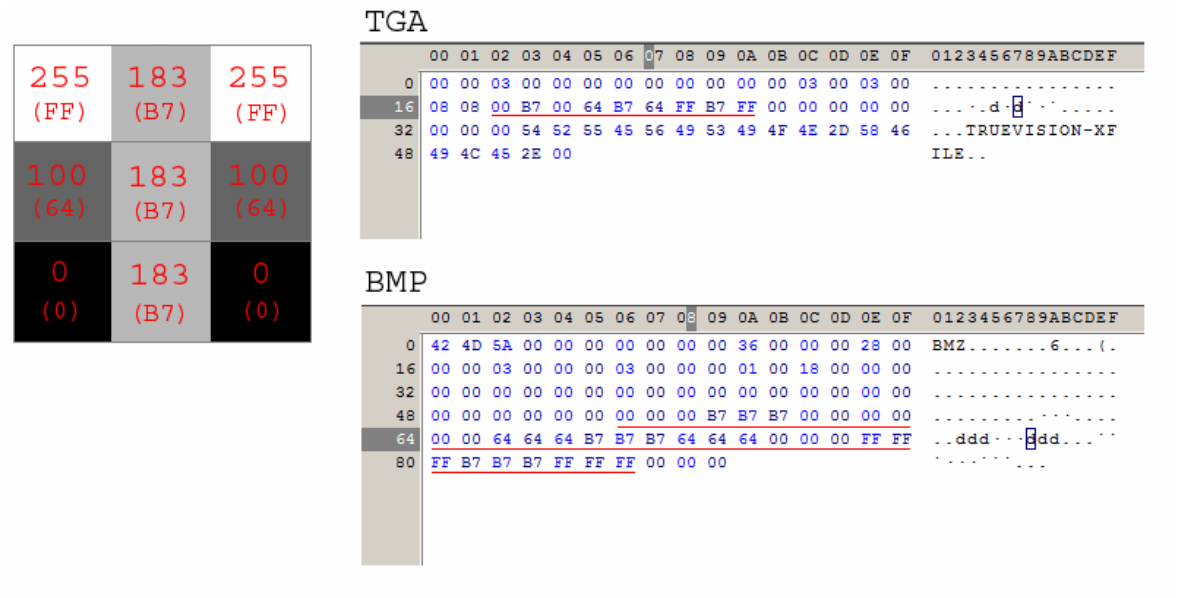
```
for(int iY = 0; iY < sHlavickaTGA.Height; iY++){ // pro každý řádek
  for(int iX = 0; iX < iDelkaRadku; iX++) radek[iX] = 0;
  for(int iX = 0; iX < sHlavickaTGA.Width*3; iX++)
  radek[iX] = pabFileData[pozice++]; // naplnění řádku
  fwrite(radek, sizeof(unsigned char), iDelkaRadku, outFile);}
```

6.2.2 TGA v odstínech šedi do 24 bitového BMP

Opět velice jednoduché. Každý obrazový pixel zastupuje jeden jediný byt – informace o jasu. Jeden pixel může tedy nabývat 256 úrovní barev, kde hodnota 255 je bílá a hodnota 0 je černá.

Převádíme do klasického 24 bitového BMP, kde je každý pixel popsán trojicí bytů, tudíž překopírujeme hodnotu jasu původního TGA hned třikrát. Výsledkem bude černobílý obraz.

Tento postup není optimální z hlediska velikosti výsledného BMP, dvě třetiny informací jsou totiž nadbytečné. Vše je však koncipováno tak, že výsledný soubor BMP je pouze prostředkem, který slouží k vykreslení obrazu a poté je odstraněn, čímž je tato nevýhoda částečně odstraněna.



Obr. 7 - Převod TGA v odstínech šedi do 24 bitového BMP

6.2.3 TGA s barevnou paletou do 24 bitového BMP

Podstatou je to, že bezprostředně za hlavičkou nenásledují obrazová data, jako tomu bylo dosud, ale je zde obsažena barevná paleta, což je 256 barev popsaných klasickými třemi byty, které prezentují červenou, zelenou a modrou barevnou složku bodu. Po paletě teprve začínají obrazová data (18 prvních bytů zabírá hlavička, trojnásobek 256 bytů zaplňuje

paleta, obrazová data tedy začínají na 786. bytu), které jsou tvořeny pouhými indexy do barevné palety. Barev je v paletě 256 a tak nám stačí pouhý jeden byt na každý pixel.

```
for(int iY = 0; iY < sHlavickaTGA.Height; iY++)
{
 for(int iX = 0; iX < iDelkaRadku; iX++)   radek[iX] = 0;
 for(int iX = 0; iX < sHlavickaTGA.Width*3;)
 {
      unsigned char hodnota = pabFileData[pozice++];
      radek[iX++] = pabFileData[zacatekPalety + 3*hodnota];
      radek[iX++] = pabFileData[zacatekPalety + 3*hodnota+1];
      radek[iX++] = pabFileData[zacatekPalety + 3*hodnota+2];
 }
 fwrite(radek, sizeof(unsigned char), iDelkaRadku, outFile);
}
```

6.2.4 24 bitový BMP do 24 bitového TGA bez RLE komprese

Tento algoritmus je součást funkce, která převádí 24 bitové BMP na 24 bitové TGA, která se používá při ukládání.

Hlavička BMP je načtena a převedena na hlavičku TGA podobně jako v předchozích případech. Vlastní algoritmus převodu vypadá následovně:

```
 for(int iY = 0; iY < sHlavickaBMP.biHeight; iY++)
 {
        for(int iX = 0; iX < iDelkaRadku; iX++)
        {
              if(iX<(sHlavickaBMP.biWidth*3))
              radek[iX] = pabFileData[pozice++];
              else pozice++;
        }
      fwrite(radek, sizeof(char), sHlavickaBMP.biWidth*3, outFile);
  }
```

Opět se jedná o velice jednoduché překopírování bytů, které nesou informaci. Je nutno pouze vynechat některé nulové byty, které mohou být v BMP zbytečně navíc.

6.2.5 24 bitový BMP do 24 bitového TGA s RLE kompresí

Princip je podobný jako v předchozím případě, algoritmus vypadá však na první pohled o mnoho složitěji.

Do výstupního souboru nemůžeme zapisovat po celých řádcích, protože předem neznáme jejich délku. Zapisovat tedy musíme po jednotlivých bytech. Hlavní část programu opět probíhá tolikrát, kolik má vstupní soubor řádků. Protože je délka řádku vstupního BMP známá a konstantní, můžeme jednoduše spočítat počáteční index každého řádku (hlavička BMP zabírá 54 bytů, pro třetí řádek je tedy index jeho počátku roven součtu 54 a dvojnásobku délky řádku). Poté následuje cyklus, který probíhá tak dlouho, dokud nezapíšeme do výstupního TGA právě tolik pixelů, kolik jich má jeden řádek. V této smyčce provedeme výpočet, který nám určí, kolik následuje stejných nebo rozdílných pixelů. Protože máme na počet opakování pouhý jeden byt a to pro jak variantu, kdy následuje sled různých bodů, tak pro variantu stejných obrazových bodů, musí být počet opakování menší než 128. Poté zapíšeme do výstupního souboru počet opakování a po něm i příslušné hodnoty.

6.2.6 24 bitový BMP do TGA s barevnou paletou

Nejdůležitější je samotná tvorba palety, která obsahuje 256 barev a měla by být vytvořena, co nejideálněji k danému obrázku.

Při psaní kódu jsem se rozhodl pro nejjednodušší variantu – pro neadaptivní metodu tvorby palety. Tato metoda vůbec neuvažuje informace obsažené v obraze, paleta je univerzální pro všechny obrázky. Nevýhodou je případná redundantnost palety, protože řada barev nemusí být ve výsledném obrázku vůbec obsažena.

Princip tvorby neadaptivní palety je takový, že každá ze tří os v prostoru RGB o délce 256 je dělena rovinami na několik částí a vzniklé průsečíky jsou vkládány do palety. Prakticky je paleta vytvořena třemi jednoduchými cykly. Kód by mohl vypadat třeba takto:

```
unsigned char *p = new unsigned char[(256*3)];
 int indexPalety = 0;

 for(int r = 0; r < 8; r++){
      for(int g = 0; g < 8; g++){
          for(int b = 0; b < 4; b++){
               p[indexPalety++]=r/7*255;
               p[indexPalety++]=g/7*255;
               p[indexPalety++]=b/3*255;
             }
         }
  }
  fwrite(p, sizeof(unsigned char), (256*3), outFile);
```

Následuje cyklus, který probíhá pro každý řádek bitmapy. Pro každý pixel hledáme nejbližší barvu z palety. Pro tento výpočet jsem vyzkoušel několik variant, ale všechny dávaly prakticky totožné výsledky, takže jsem nakonec použil nejjednodušší metodu – sečítání absolutní hodnoty rozdílů složek aktuálního pixelu a hodnot v paletě. Nejlepší přiblížení, tedy nejnižší součet rozdílů, je uchováván společně s patřičným indexem. Takhle je procházena celá paleta a nejlepší výsledek je zapsán do výstupního souboru.

Použitý postup není nejideálnější, ale je spolehlivě funkční, relativně rychlý a jednoduchý. Hlavním vylepšením by bylo použití adaptivní, heuristické nebo iterativní metody tvorby palety.

6.2.7 24 bitový PCX do 24 bitového BMP

Nejprve je načtena hlavička PCX a převedena na hlavičku BMP stejně jako ve všech ostatních případech. Práce se soubory PCX je složitější a nepřehlednější v tom, že data se ukládají ve formátu $R_0R_1R_2R_3R_4G_0G_1G_2G_3G_4B_0B_1B_2B_3B_4$ a obraz se zapisuje v obráceném pořadí než je tomu u TGA nebo BMP, totiž zapisuje se odshora dolů.

RLE komprese je poněkud rozdílná – opakují se stejné byty a ne shodné pixely jako u TGA. Formát PCX může obsahovat zarovnávací nuly podobně jako BMP (jeden řádek musí být totiž reprezentován sudým počtem bytů) a i tyto znaky mohou být zahrnuty i v RLE kompresi.

6.2.8 PCX s barevnou paletou do 24 bitového BMP

Formát PCX může obsahovat 256 barevnou paletu, její použití je však poměrně problematické. Paleta se umísťuje na samotný konec souboru, přičemž samotnou přítomnost palety není úplně snadné určit. Přítomnost palety značí byt, který je mezi paletou a samotnými obrazovými daty, pokud je jeho hodnota 12, paleta je přítomna. Není však zaručeno, že právě v tomto bytu není zapsaná tato hodnota, i když se jedná o jiný typ souboru.

Samotný princip je totožný s TGA variantou, obrazová data tvoří indexy do barvové palety. Každý pixel je tudíž popsán jedním jediným bytem.

6.3 Doplňkové funkce

Knihovna dále obsahuje další funkce pro práci s BMP souborem. Jedná se o dva druhy převrácení a o inverzi barev.

6.3.1 Vertikální převrácení

Jedná se o jednoduché zapsání řádků v opačném pořadí.

```
int i = 0;
unsigned char *radek = new unsigned char[iDelkaRadku];
do{
        i++;
        poz = clFileSize - i*iDelkaRadku;
        for(int idx = 0; idx < iDelkaRadku; idx++)
              radek[idx] = pabFileData[poz + idx];
        fwrite(radek, sizeof(unsigned char), iDelkaRadku, outFile);
   }while(i < sHlavickaBMP.biHeight);
```

6.3.2 Horizontální převrácení

Převracíme každý řádek zvlášť, zapisujeme trojice bytů v opačném pořadí, nesmíme zapomenout na zarovnávací nuly, které nenesou žádnou informaci a mohou být v řádku obsaženy.

```
for(int iY=0; iY < sHlavickaBMP.biHeight; iY++){ // pro každý řádek
 int pozice = 54 + (iY * iDelkaRadku);
 for(int idx = 0; idx < iDelkaRadku; idx++)     radek[idx] = 0;
 for(int idx = 0; idx < sHlavickaBMP.biWidth*3; idx++)
 radek[idx] = pabFileData[(pozice+(sHlavickaBMP.biWidth*3)-idx-1)];
 // převedeno na tvar BGRBGRBGR
 for(int idx = 0; idx < sHlavickaBMP.biWidth*3; idx=idx+3){
       char B = radek[idx];    char R = radek[idx+2];
       radek[idx] = R;         radek[idx+2] = B;
 } // převedeno na tvar RGBRGBRGB
 fwrite(radek, sizeof(unsigned char), iDelkaRadku, outFile);}
```

6.3.3 Inverze barev

Každý řádek procházíme byt po bytu a invertujeme hodnoty (nová hodnota je rovna rozdílu čísla 255 a původní hodnoty bytu), nesmíme opomenout zarovnávací byty, které musejí zůstat nulové.

```
for(int iY = 0; iY < sHlavickaBMP.biHeight; iY++){
      int pozice = 54 + (iY * iDelkaRadku);
      for(int idx = 0; idx < iDelkaRadku; idx++) radek[idx] = 0;

      for(int iX = 0; iX < sHlavickaBMP.biWidth*3; iX++)
      radek[iX] = 255 - pabFileData[pozice + iX];

      fwrite(radek, sizeof(unsigned char), iDelkaRadku, outFile);
}
```

Kompletní výčet knihovních funkcí je v příloze P I – Souhrn funkcí.

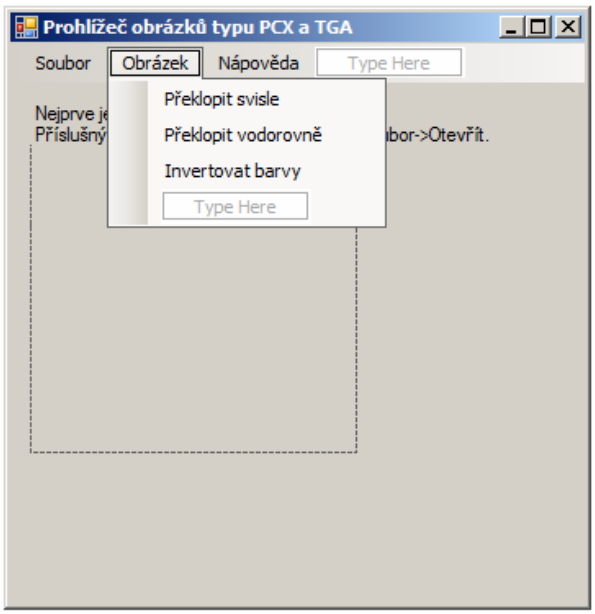
6.4 Implementace v jiných programech

Díky důslednému oddělení knihovny a vlastního programu je implementace velice jednoduchá. Hlavičkový soubor obsahuje i celá těla funkcí, tudíž stačí zkopírovat soubor konvertor.h do adresáře s programem a ve vlastním programu jej vložit příkazem #include. Poté je již možno volat příslušné funkce.

7 PROGRAM

7.1 Windows Forms aplikace

Dále byla vytvořena aplikace, která využívá knihovny, jejíž popis je uveden výše. Rozhodl jsem se pro jednoduchou Windows Forms aplikaci v prostředí Visual Studio 2008. Nevýhodou je nutnost instalace balíčku .NET Framework na cílový počítač.



Obr. 8 - Windows Forms aplikace

7.1.1 Práce se soubory z programátorského hlediska

Před načtením obrázku je nutno zvolit pracovní složku, do které bude ukládán soubor BMP, jenž je výstupem knihovních funkcí. Dialog se otevře automaticky po kliknutí na volbu Otevřít. Poté se teprve otevře dialog, který slouží k vlastnímu otevření souboru. Po vybrání obrázku je spuštěna patřičná knihovní funkce (PCXtoBMP pro formát PCX a TGAtoBMP pro formát TGA, jedná se o funkce, které z hlavičky vyčtou konkrétní variantu bitmapy a zavolají příslušnou funkci). Na místě, které jsme vybrali, je vytvořen soubor prac.bmp, jenž je automaticky vymazán při vypnutí programu. Vytvořené BMP je jednoduchým způsobem zobrazeno pomocí objektu pictureBox.

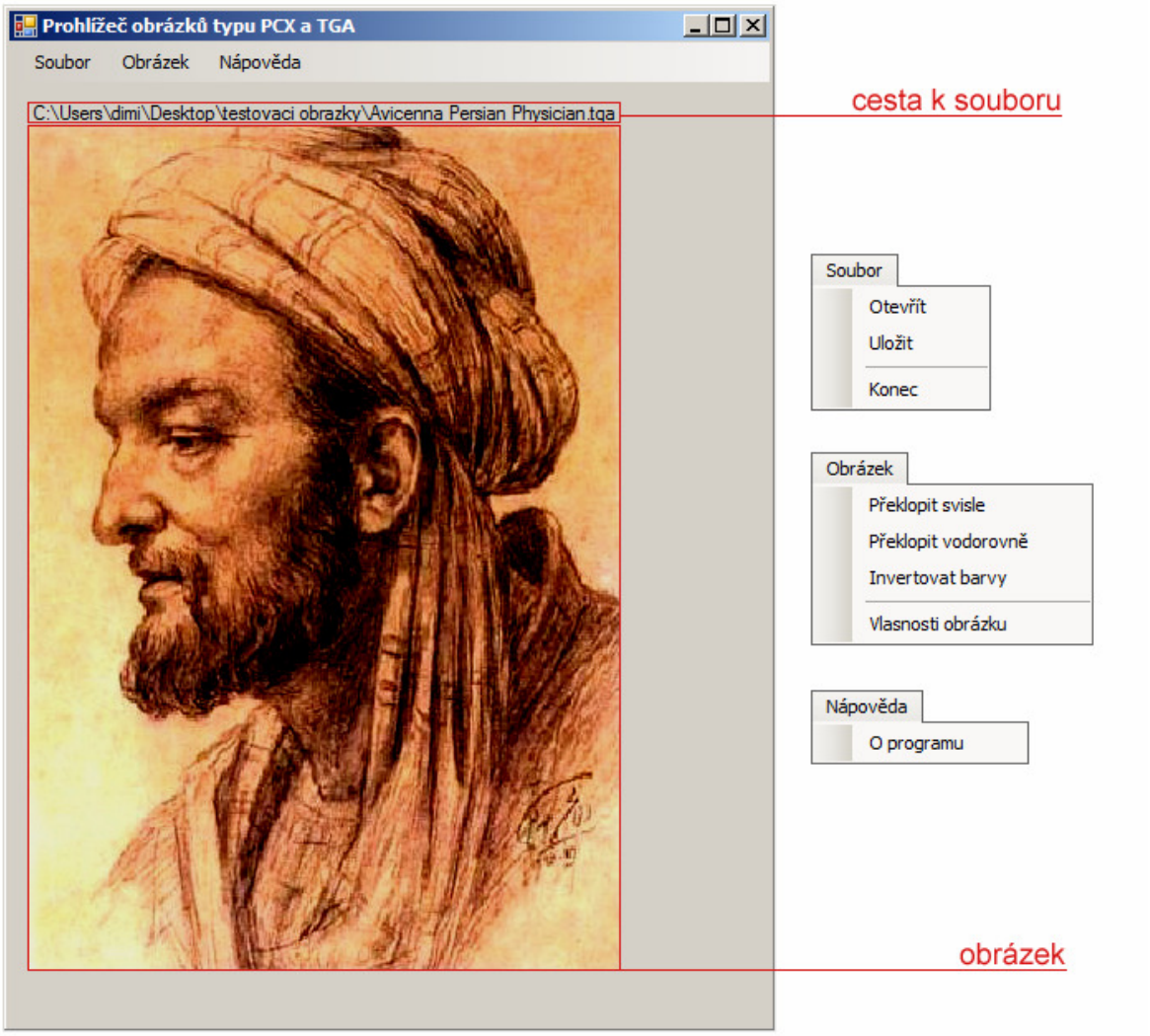
```
img = System::Drawing::Bitmap::FromFile(BMPfile);
System::Drawing::Bitmap^ img2;
img2 = gcnew Bitmap(img);
pictureBox1->Image = img2;
delete img;
```

Při zobrazování PCX je situace mírně složitější, protože výsledný obraz je převrácen „vzhůru nohama“. Je potřeba jej převrátit pomocí funkce ReverseBMP.

```
PCX24toBMP24(strOPEN, strBMP);
ReverseBMP(strBMP, strBMPR);
remove(strBMP);
rename (strBMPR, strBMP);
```

Při ukládání je jednoduše volána funkce, která převádí BMP na příslušný formát.

7.2 Uživatelská příručka



Obr. 9 - Program

Soubor – Otevřít

Tato volba slouží k načtení obrázku, při prvním načítání se po stisknutí tlačítka zobrazí dialog, pomocí kterého je možno zvolit pracovní složku – místo, kam se budou ukládat přechodné soubory. V této složce je třeba mít potřebná práva ke čtení a zápisu. Po zvolení pracovní složky se zobrazí další dialogové okno – to slouží k otevření samotného bitmapového souboru. Je možno otvírat soubory typu PCX a TGA, typ souboru je možno zvolit v rozevírací nabídce. Po zvolení souboru je obrázek zobrazen v hlavní části okna, nad obrázkem se vypíše cesta k souboru. Po načtení je možno s obrázkem dále pracovat – lze provádět ukládání, inverzi barev, horizontální a vertikální převrácení.

Soubor – Uložit

Pokud je načtena a zobrazena bitmapa, můžeme ji uložit. Po kliknutí na tuto volbu se zobrazí klasické dialogové okno Uložit jako. Do sekce Název souboru je nutno napsat název výsledné bitmapy. Pomocí rozevírací nabídky Uložit jako typ je možno zvolit typ výsledného souboru. Ukládat lze do plnobarevného PCX, plnobarevného TGA s kompresí, plnobarevného TGA bez komprese, do TGA v 256 barvách a do TGA v odstínech šedé. Poslední dvě varianty provedou redukci barev.

Soubor – Konec

Po kliknutí na tuto volbu jsou smazány pomocné soubory v pracovní složce, jsou-li nějaké, a poté je program ukončen.

Obrázek – Překlopit svisle

Provede svislé překlopení bitmapy.

Obr. 10 - Svislé překlopení

Obrázek – Překlopit vodorovně

Provede vodorovné překlopení bitmapy.

Obr. 11 - Vodorovné překlopení

Obrázek – Invertovat barvy

Provede inverzi barev.

Obr. 12 - Inverze barev

Obrázek – Vlastnosti obrázku

Zobrazí základní vlastnosti obrázku – typ souboru, bitovou hloubku a rozměry bitmapy.

Nápověda – O programu

Po kliknutí jsou zobrazeny základní informace o programu.

8 PREZENTACE PRO VÝUKU

V programu Microsoft PowerPoint byla vytvořena prezentace pro potřeby výuky předmětu Počítačová grafika. Jsou v ní popsány grafické formáty PCX a TGA, které jsou podrobněji rozebrány v teoretické části bakalářské práce.

Na obr. 13 je ukázková dvojice snímků z této prezentace.

Obr. 13 - Ukázka prezentace

ZÁVĚR

Cílem bakalářské práce bylo vytvořit literární rešerši, zabývající se historií a vývojem rastrových grafických formátů PCX a TGA. Dalším bodem zadání bylo naprogramovat jednoduchou aplikaci zobrazující uvedené formáty a uzpůsobit zdrojové kódy tak, aby je bylo možno implementovat v jiné aplikaci.

V teoretické části jsem se nejprve věnoval obecnému popisu bitmapové grafiky. Shrnul jsem její výhody, nevýhody a praktické využití. Prostor jsem věnoval krátkému popisu několika typických bitmapových formátů. Poté jsem detailně popsal historii a strukturu formátů PCX a TGA. Popsal jsem vnitřní strukturu těchto formátů, členění souborů, možnosti uplatnění a jejich výhody a nevýhody.

V praktické části jsem vyvinul aplikaci zobrazující oba formáty. Kromě načítání, ukládání a zobrazování program obsahuje také několik jednoduchých funkcí, jako je např. horizontální a vertikální převrácení nebo inverze barev. Aplikace má jednoduché grafické rozhraní a intuitivní ovládání. Program jsem úspěšně otestoval na operačních systémech Windows XP a Windows Vista. Nevýhodou je nutnost nainstalování balíčku Microsoft .NET Framework na cílový počítač.

Program je důsledně rozdělen na dvě části – knihovnu pro práci s obrázky a vlastní grafické rozhraní. Díky tomu je možno funkce pro práci se soubory typu PCX a TGA velice jednoduše využít i v jiných programech a aplikacích. Program je řádně popsán, což usnadňuje pochopení zdrojového kódu.

Na přiloženém CD je umístěn program i s kompletními zdrojovými soubory a prezentací pro potřeby výuky předmětu Počítačová grafika.

ZÁVĚR V ANGLIČTINĚ

The aim of the bechelor's thesis is to create the literature search, which is interested in the history and the progress of the raster graphics formats PCX and TGA. Another point was a simple application showing these formats and adapt the source code so that it can be implemented into other applications.

In the theoretical part, I first focused on the general description of bitmap graphics. I summed up the benefits and use. I wrote a brief description of several typical bitmap formats. Then I described in detail the history and structure of PCX and TGA formats. I described the internal structure of these formats, broken files, application possibilities and their advantages and disadvantages.

In the practical part, I developed an application displaying both formats. In addition to saving, fetching and displaying program also contains a few simple functions such as horizontal and vertical overturning and color inversion. The application has simple graphics interface – line and intuitive operating. The program was successfully tested on Windows XP and Windows Vista. The disadvantage is the need to install the Microsoft package. NET Framework on the target computer.

The program is strictly divided into two parts - a library for working with images and graphical interface. The program is correctly described for easy understanding of the source code.

On the CD enclosed to this thesis is a program with complete source files and presentation for teaching the subject of Computer Graphics.

SEZNAM POUŽITÉ LITERATURY

[1] LIBERTY, Jesse. Naučte se C++ za 21 dní. Brno : Computer Press, 2007. 796 s. ISBN 978-80-251-1583-1.

[2] MURRAY, James D., VANRYPER, William. Encyklopedie grafických formátů. 1. vyd. Praha : Computer Press, 1997. 922 s. ISBN 80-7226-033-2.

[3] TIŠNOVSKÝ, Pavel. Grafický formát PCX - výlet do historie PC [online]. 2006 [cit. 2008-11-30]. Dostupný z www: <http://www.root.cz/clanky/graficky-format-pcx-vylet-do-historie-pc/#k08>.

[4] TIŠNOVSKÝ, Pavel. Grafický formát TGA - jednoduchý, oblíbený, používaný [online]. 2006 [cit. 2008-11-30]. Dostupný z www: <http://www.root.cz/clanky/graficky-format-tga-jednoduchy-oblibeny-pouzivany>.

[5] TIŠNOVSKÝ, Pavel. PCX prakticky - implementace komprimace RLE [online]. 2006 [cit. 2009-01-24]. Dostupný z www: <http://www.root.cz/clanky/pcx-prakticky-implementace-komprimace-rle/>.

[6] TIŠNOVSKÝ, Pavel. Grafický formát TGA prakticky [online]. 2006 [cit. 2009-01-24]. Dostupný z www: <http://www.root.cz/clanky/graficky-format-tga-prakticky/>.

[7] ŽÁRA, Jiří, BENEŠ, Bedřich, FELKEL, Petr. Moderní počítačová grafika. 1. vyd. Praha : Computer Press, 1998. 448 s. ISBN 80-7226-049-9.

[8] Bitmapová grafika [online]. 2008 [cit. 2008-11-23]. Dostupný z www: <http://www.symbio.cz/slovnik/bitmapova-grafika.html>.

[9] Bitmapová grafika [online]. 2008 [cit. 2008-11-23]. Dostupný z www: <http://www.adaptic.cz/znalosti/slovnicek/bitmapova-grafika.htm>.

[10] Rastrová grafika [online]. 2008 [cit. 2008-11-23]. Dostupný z www: <http://cs.wikipedia.org/wiki/Bitmapov%C3%A1_grafika>.

[11] Raster Graphics [online]. 2008 [cit. 2008-11-23]. Dostupný z www: <http://en.wikipedia.org/wiki/Raster_graphics/>.

[12] Rastrové formáty [online]. 2004 [cit. 2008-11-23]. Dostupný z www: <http://nlp.fi.muni.cz/nlp/NlpCz/Rastrove_formaty.html>.

[13] Komprimační algoritmy [online]. 2008 [cit. 2008-11-30]. Dostupný z www: <http://gimli.mysteria.cz/komprese/algoritmy.html#rle>.

[14] Endianita [online]. 2008 [cit. 2009-01-24]. Dostupný z www: <http://cs.wikipedia.org/wiki/Endianita/>.

[15] Microsoft Visual Studio [online]. 2009 [cit. 2009-04-09]. Dostupný z www: <http://cs.wikipedia.org/wiki/Microsoft_Visual_Studio>.

[16] Visual Studio, domovská stránka [online]. 2009 [cit. 2009-04-24]. Dostupný z www: < http://www.microsoft.com/cze/msdn/produkty/vstudio/default.mspx>.

[17] MiTec, domovská stránka [online]. 2009 [cit. 2009-04-24]. Dostupný z www: <http://www.mitec.cz/hex.html>.

[18] PCX [online]. 2009 [cit. 2009-01-24]. Dostupný z www: <http://en.wikipedia.org/w/index.php?title=PCX&oldid=257586417>.

[19] Truevision TGA [online]. 2009 [cit. 2009-01-24]. Dostupný z www: <http://en.wikipedia.org/w/index.php?title=Truevision_TGA&oldid=263137981>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

BMP Microsoft Windows Bitmap nebo také .DIB (device-independent bitmap) je počítačový formát pro ukládaní rastrové grafiky.

C Programovací jazyk ANSI C.

C++ Programovací jazyk ISO C++ vzešlý z jazyka ANSI C.

CMYK Barevný model založený na subtraktivním míchání barev.

DTP Jedná se o tvorbu tištěného dokumentu za pomoci počítače.

EGA Enhanced Graphic Adapter je standard grafické karty.

GIF Graphics Interchange Format je grafický formát určený pro rastrovou grafiku.

IBM International Business Machines Corporation je přední světová společnost v oboru informačních technologií.

JPG Standardní metoda ztrátové komprese používané pro ukládání počítačových obrázků. Formát, který tuto kompresi používá, se také běžně nazývá JPG.

LZW Bezeztrátový kompresní algoritmus vyvinutý Abrahamem Lempelem, Jacobem Zivem a Terry Welchem.

PCX PC Paintbrush File Format - počítačový souborový formát firmy ZSoft pro ukládaní rastrové grafiky.

PNG Portable Network Graphics je grafický formát určený pro bezeztrátovou kompresi rastrové grafiky.

RGB Barevný model založený na aditivním míchání barev.

RGBA Barevný model, který vychází z modelu RGB, je rozšířen o tzv. alfa kanál.

RLE Run Length Encoding je bezeztrátová komprese, která kóduje vstupní data tak, že kóduje posloupnosti stejných hodnot do dvojic (délka posloupnosti, hodnota).

TGA Targa - jeden ze souborových formátů pro ukládání rastrové počítačové grafiky.

VGA Video Graphics Array je počítačový standard pro počítačovou zobrazovací techniku, vydaný roku 1987 společností IBM.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA PI: SOUHRN FUNKCÍ

BMP24toPCX24(FileBMP, FilePCX)	Převádí plnobarevný BMP na plnobarevný PCX.
BMP24toPCX24Pal(FileBMP, FilePCX)	Převádí plnobarevný BMP na PCX s barevnou paletou.
PCX24toBMP24(FilePCX, FileBMP)	Převádí plnobarevný PCX na plnobarevný BMP.
PCX24PaltoBMP24(FilePCX, FileBMP)	Převádí PCX s barevnou paletou na plnobarevný BMP.
PCXtoBMP(FilePCX, FileBMP)	Funkce načítá hlavičku PCX a poté volá příslušnou funkci pro převod.
BMP24toTGA24(FileBMP, FileTGA)	Převádí plnobarevný BMP na plnobarevný TGA bez RLE komprese.
BMP24toTGA24Gray(FileBMP, FileTGA)	Převádí plnobarevný BMP na TGA v odstínech šedi.
BMP24toTGA24RLE(FileBMP, FileTGA)	Převádí plnobarevný BMP na plnobarevný TGA s RLE kompresí.
BMP24toTGA24Pal(FileBMP, FileTGA)	Převádí plnobarevný BMP na TGA s barevnou paletou.
TGA24toBMP24(FileTGA, FileBMP)	Převádí plnobarevný TGA bez RLE komprese na plnobarevný BMP.
TGA24RLEtoBMP24(FileTGA, FileBMP)	Převádí plnobarevný TGA s RLE kompresí na plnobarevný BMP.
TGA24PaltoBMP24(FileTGA, FileBMP)	Převádí TGA s barevnou paletou na plnobarevný BMP.
TGA24GraytoBMP24(FileTGA, FileBMP)	Převádí TGA v odstínech šedi na plnobarevný BMP.
TGAtoBMP(FileBMP, FileTGA)	Funkce načítá hlavičku TGA a poté volá příslušnou funkci pro převod.
ReverseBMP(FileIn, FileOut)	Převrátí obrázek vertikálně.
ReverseIIBMP(FileIn, FileOut)	Převrátí obrázek horizontálně.
InverseBMP(FileIn, FileOut)	Provede inverzi barev.