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

VÝUKOVÝ PROGRAM PRO DEMONSTRACI METOD REDUKCE BAREVNÉHO PROSTORU OBRAZŮ

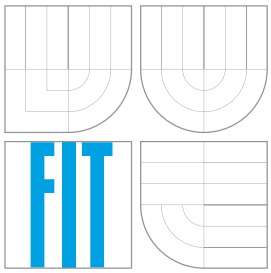
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DANIEL PEKAŘ

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

VÝUKOVÝ PROGRAM PRO DEMONSTRACI METOD REDUKCE BAREVNÉHO PROSTORU OBRAZŮ

EDUCATION COMPUTER PROGRAM FOR DEMONSTRATION OF IMAGES COLORS SPACE REDUCTION METHODS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — DANIEL PEKAŘ
AUTHOR

VEDOUCÍ PRÁCE — Ing. VÍT ŠTANCL
SUPERVISOR

BRNO 2007

Zadání bakalářské práce/5338/2006/xpekar04

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Pekař Daniel**
Obor: Informační technologie
Téma: **Výukový program pro demonstraci metod redukce barevného prostoru obrazů**
Kategorie: Počítačová grafika
Pokyny:

1. Prostudujte problematiku redukce barevného prostoru obrazů v počítačové grafice.
2. Prostudujte problematiku tvorby výukových a demonstračních programů.
3. Navrhněte výukový program pro demonstraci metod redukce barevného prostoru obrazů v počítačové grafice.
4. Implementujte navržený program ve vybraném jazyce (C/C++, Java, Python, C#).
5. Ve vytvořeném programu implementujte popsané principy a barevné modely.

Literatura:

- Žara J., Beneš B., Felkel P.: Moderní počítačová grafika. 1. vyd. Praha, Computer press 1998, 448 s., ISBN 80-7226-049-9

Při obhajobě semestrální části projektu je požadováno:

- Splnění prvních 3 bodů zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Štancl Vít, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Pavel Zemčík
vedoucí ústavu

Licenční smlouva

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

Abstrakt

Práce popisuje tvorbu výukového programu pro demonstraci metod redukce barevného prostoru obrazů. Program neslouží pouze jako nástroj na redukování barev v obrázcích, ale poukazuje také na princip jednotlivých redukčních metod. Možnosti využití programu se rozšiřují díky spojení teoretických východisek s praktickou aplikací. Může sloužit jako nástroj pro demonstrování redukčních metod při výuce nebo také jako studijní materiál pro uživatele při samostudiu.

Klíčová slova

Barva, RGB, CMY, rozptylování, polotónování, stupně šedi, náhodné rozptýlení, prahování, distribuce chyby, maticové rozptylování, maticové polotónování.

Abstract

This work describes an Education Computer Program for Demonstration of Images Colors Space Reduction Methods creation. This program does not only constitute a means to reduce colors space in images but adverts to principles of individual reduction methods as well. The possibilities of program´s employment expand due to theoretical background´s incorporation into a practical application. The program can be in service as a reduction methods´demonstration tool in a class or as a study material for an individual learner.

Keywords

Color, RGB, CMY, dithering, halftoning, gray scale, random dithering, thresholding, error diffusion, matrix dithering, matrix halftonig.

Citace

Daniel Pekař: Výukový program pro demonstraci metod redukce barevného prostoru obrazů, bakalářská práce, Brno, FIT VUT v Brně, 2007

Výukový program pro demonstraci metod redukce barevného prostoru obrazů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Víta Štancla

.....

Daniel Pekař

14.5.2007

Obsah

Kapitola 1

Úvod

Cílem mé bakalářské práce bylo vytvořit výukový program, který demonstruje metody redukce barevného prostoru obrazů. V zadání není přesně specifikováno, zda program budou požívat učitelé či studenti. Z tohoto důvodu musí být univerzální. Učitel prostřednictvím této aplikace bude moci demonstrovat jednotlivé metody redukce a ukazovat rozdíly mezi nimi a student si k tomu bude moci zobrazit ještě teorii, která popisuje činnost redukčních algoritmů.

Práci jsem rozdělil do osmi kapitol. Hned po úvodu následuje kapitola, která objasní základní pojmy o světle a barvě a přiblíží princip, jakým se barva zobrazuje v počítačové grafice. Třetí kapitola je již věnována redukci barev. Jsou zde popsány nejčastěji používané redukční metody. Zbývající část práce je zaměřena na tvorbu samotné aplikace. Nejprve jsou sepsány požadavky na výukový program. Ty najdeme ve čtvrté kapitole. Následuje analýza spolu s objektovým návrhem aplikace. V podkapitolách této části se nachází bližší popis jednotlivých tříd a návrh komunikace mezi nimi. Kapitola šest obsahuje popis řešení jednotlivých částí programu, tedy implementaci. V ní se zmiňuji o různých problematikách a způsobu jejich řešení. Předposlední kapitola popisuje práci s programem. A poslední osmá kapitola je závěr. V závěru je shrnutí celé práce a také možnosti jejího rozšíření či vylepšení.

Kapitola 2

Teorie barev

2.1 Světlo a barva

Světlo je elektromagnetické záření, přičemž viditelné světlo pro člověka se pohybuje v úzkém pásmu elektromagnetického spektra v oblasti od 380 nm do 700 nm. Každá vlnová délka v tomto viditelném spektru odpovídá určité barvě. Lidské oko dokáže rozlišit přibližně 400 000 různých barev. Červené barvě odpovídá vlnová délka okolo 700 nm, přičemž na druhém konci viditelného spektra je barva fialová s vlnovou délkou asi 380 nm.

Za zdroj světla považujeme slunce. Ale může jím být také třeba žárovka. Žádoucí je totiž, aby zdroj světla vysílal velké množství fotonů, jejichž vlnové délky odpovídají viditelné části spektra. Ty se skládají tak, že tvoří bílé světlo. Takové světlo se nazývá *achromatické*. Předmět, na něhož světlo dopadá, některé frekvence pohltí a jiné odrazí. A právě odražené světlo obsahuje frekvence, které vnímáme jako barvu. Podle toho, která frekvence převládá (je dominantní), určujeme barvu, neboli *barevný tón* předmětu. Dopadá-li na povrch předmětu jiná intenzita světla, například pokud je šero, jas odraženého světla je nižší (předmět je jakoby tmavší). Další pojem související se světlem a barvou, je *sytost*. Ta udává čistotou barvy světla. Čím je ve světle užší spektrum frekvencí, tím vyšší je sytost barvy. Posledním pojmem, který zmíním je *světlost* barvy. Určuje ji velikost achromatické složky ve světle s určitou dominantní frekvencí. Bližší informace o barvách nebo světle lze nalézt v literatuře ze které jsem čerpal ([7] případně [3]).

2.2 Barvy v počítačové grafice

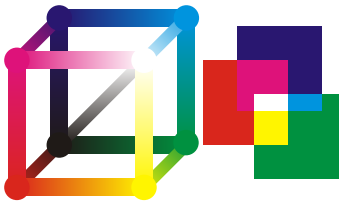
Mezinárodní komise pro osvětlení *CIE* (*Commission internationale de l'éclairage*), jak je psáno na Wikipedii [5], vytvořila tzv. *Chromatický diagram CIE*. Jsou do něj vyneseny všechny barvy viditelného spektra. Křivka diagramu připomíná podkovu. Pro počítačovou grafiku je důležitá jedna jeho základní vlastnost. Vybereme-li libovolné tři body, které leží uvnitř diagramu, a neleží v přímce (tedy tvoří trojúhelník), můžeme jejich mícháním vytvořit libovolnou barvu, která leží uvnitř trojúhelníku. Další informace o CIE jsou k nalezení například v [4].

V praxi samozřejmě chceme, aby trojúhelník byl co největší, tedy aby bylo možné zobrazit co nejvíce barev. Proto se snažíme umístit vrcholy co nejblíže okrajům křivky diagramu. Barvy ovšem musí být takové, abychom je mohli zobrazit. Například fosfor v CRT monitorech nedosahuje stejné barvy jako okrajové barvy v chromatickém diagramu. S tímto problémem, tedy vybrat tři vhodné barvy, souvisí pojmy barevné modely **RGB** a **CMY**.

2.2.1 Barevný model RGB

Základem barevného modelu RGB jsou tři barvy. **R**ed, **G**reen a **B**lue, tedy červená, zelená a modrá. Model je možné dobře znázornit pomocí jednotkové krychle, kde jednotlivé osy tvoří barvy r, g a b. Vše je znázorněno na obrázku 2.1. Počátek souřadného systému je černá barva a protější vrchol je bílá o souřadnicích $[1, 1, 1]$. Na této diagonále leží stupně šedi. K vytvoření dostatečného počtu barev se v počítačové grafice používá dělení intervalu na 256 dílů. Tedy interval není od 0 po 1, ale od 0 po 255. Barvu lze jednoduše zakódovat do osmi bitů. K zakódování libovolného bodu uvnitř krychle tedy postačí 24 bitů, což je bráno jako *true colors*.

RGB model je používán pro zobrazování barev na monitorech, displejích či v projektorech. Je to dáno tím, že míchání barev u RGB modelu je *aditivní*. To znamená, že jde o aktivní vyzařování barev, které nepotřebuje vnější osvětlení. Smíchání těchto tří základních barev dostaneme barvu bílou. Aditivní míchání barev i jednotková krychle jsou na obrázku 2.1.

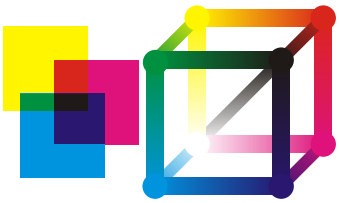


Obrázek 2.1: Jednotková krychle RGB a příklad aditivního míchání barev.

2.2.2 Barevný model CMY

Model CMY je používám především při tisku. Jde totiž o *subtraktivní* míchání barev. Barevný model CMY je podobný jako RGB. Základ opět tvoří tři základní barvy. A to **C**yan, **M**agenta a **Y**ellow, tedy azurová, purpurová a žlutá. Jednotková krychle je doplňkem k jednotkové krychli pro RGB. Pro názornost opět poslouží obrázek 2.2.

Smícháním základních tří dostaneme černou. Tedy abych byl přesný, tak je to tmavě šedá nebo tmavě hnědá. Je to způsobeno nedokonalostí základních barev c, m a y. Proto se v praxi používá model CMYK, kde písmeno **K** znamená Blac**K** nebo také **K**ey. Černé plochy jsou tisknuty jako černé za použití černé barvy. Tahle varianta je také podstatně levnější. Subtraktivní míchání barev je zobrazeno na obrázku 2.2.



Obrázek 2.2: Příklad subtraktivního míchání barev a jednotková krychle CMY.

Kapitola 3

Teorie redukce barev

Jak již bylo psáno v předchozí kapitole, barva se v počítačové grafice vytváří složením několika základních barev. Například u barevného modelu RGB je každá barevná složka tvořena 8 bity, celkem tedy 24 bitů. Díky tomu dokážeme zobrazit přibližně 16,7 mil. barev. Ovšem ne každé zobrazovací zařízení dokáže toto množství barev zobrazit. Některé monitory či displeje mohou být černobílé, nebo například obsahují pouze 256 barev. Taktéž tiskárny často bývají pouze černobílé. A právě díky těmto důvodům je třeba redukovat barevný prostor. Nežádoucím jevem při redukci barev je ztráta informace. Proto je na redukční metody kladen vysoký důraz, aby tuto ztrátu minimalizovaly. Princip většiny metod je založen na nedokonalosti lidského oka, které vnímá několik blízkých barevných bodů jako jeden. Závisí to hlavně na velikosti bodů a vzdálenosti pozorovatele. Díky této iluzi dokáží redukční metody vytvořit velkou škálu barevných odstínů prostřednictvím několika málo barev. Většinu následujících informací jsem čerpal z [7].

Všechny následující kapitoly popisují jednotlivé metody redukce barevného prostoru. Pro jednoduchost a větší názornost obrázků jsou popisovány metody pracující pouze s obrázky ve stupních šedi. Zdrojový obrázek obsahuje 256 stupňů šedi a zredukovaný pouze černou a bílou barvu (dále jen černobílý). Výjimku tvoří metoda *Převod na stupně šedi*. Tato metoda pracuje s barevným zdrojovým obrázkem. Pochopitelně i pro ostatní metody lze použít barevný zdrojový obrázek a aplikovat danou metodu na každý barevný kanál zvlášť. Ale názornější a lépe pochopitelné je použití zdrojového obrázku pouze ve stupních šedi.

3.1 Převod na stupně šedi

Mezi základní a často používané metody patří převod obrazu na obraz ve stupních šedi. Princip je jednoduchý. Pro každý pixel obrazu se spočítá jeho intenzita. Buď se použije vzorec 3.1 nebo přesněji 3.2, ve kterém se již přihlíží na citlivost vnímání lidského oka a výsledek tak vypadá věrohodněji. Tato výsledná intenzita je pak přiřazena do všech třech barevných složek původního pixelu. Nejčastěji se používá 8 bitové vzorkování, výsledný obraz potom obsahuje 256 úrovní šedi. Tento počet je dostačující, protože lidské oko dokáže rozeznat maximálně asi 60 úrovní šedé. Příklad je na obrázku 3.1.

$$I = 1/3 * (R + G + B) \tag{3.1}$$

$$I = 0,299R + 0,587G + 0,114B \tag{3.2}$$

Obrázek 3.1: Příklad převodu barevného obrázku na obrázek s 256 odstíny šedé.

3.2 Halftoning a Dithering

Tyto dva pojmy souvisí s metodami převodu obrazu o 256 stupních šedé na černobílý. Základní rozdíl je ve velikosti (tedy v rozlišení) výsledného obrazu. Metody spolu souvisí tak blízce, že halftoning můžeme chápat jako speciální případ ditheringu, jak se uvádí v některých literaturách.

Při **ditheringu** (rozptylovaní) je každý pixel obrazu nahrazen novou hodnotou podle zvolené metody. Výsledný obraz tedy nezmění své rozlišení a obsahuje jen pixely černé a bílé. Ditheringu se využívá v monitorech nebo displejích s omezeným počtem barev.

Naopak při **halftoningu** (polotónování) je každý pixel nahrazen několika novými pixely černé nebo bílé barvy. Rozlišení výsledného obrazu se tak zvětší v závislosti na tom, kolika pixely byl původní nahrazen. Halftoning se uplatňuje především v tiskárnách. Ty totiž pracují s daleko vyšším rozlišením než monitory, proto zvětšení výsledného obrazu není na škodu.

3.3 Náhodné rozptýlení

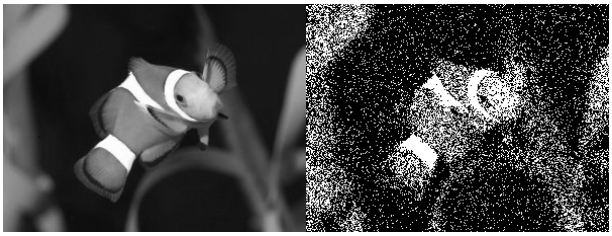
Jedna z nejjednodušších metod převodu obrazu o 256 barvách na černobílý. Metoda patří mezi rozptylovací, tedy spadá pod **dithering**. Každý pixel obrazu je porovnán s náhodným číslem v intervalu $< 0, 255 >$. V případě, že intenzita původního pixelu je nižší než náhodná hodnota, tak nový pixel bude černý, v opačném případě bílý. Výsledný obraz není příliš kvalitní, avšak jasové poměry zůstávají zachovány. Pro správnou funkčnost této metody musíme používat vhodný generátor pseudonáhodných čísel. V případě nevhodného generátoru se mohou ve výsledném obrazu objevit nežádoucí jevy. Příklad náhodného rozptýlení je na obrázku 3.2. Algoritmus metody je následující.

Pro všechny pixely obrazu:

- Inicializace $C_{OUT} = 0$
- Je-li $C_{IN} >$random(C_{MAX}),
  pak $C_{OUT} += 1$

3.4 Prahování

Velice podobná metoda jako náhodné rozptýlení. Taktéž spadá pod **dithering** a je velice jednoduchá (nejjednodušší z uvedených metod). Základem je hodnota prahu (*threshold*),



Obrázek 3.2: Příklad převodu obrázku o 256 stupních šedé na obrázek černobílý pomocí náhodného rozptýlení.

podle které se určuje nová hodnota pixelu. V případě, že zpracovávaný pixel má nižší intenzitu než je hodnota prahu, nový pixel je černý, v opačném případě bílý. Práh může být libovolný v rozmezí intenzity pixelu, tedy $< 0, 255 >$. Nejčastěji se používá polovina intervalu (128), použitelné jsou ale také střední hodnota nebo medián. Příklad je na obrázku 3.3. Pro obrázek byla použita hodnota prahu 128. Algoritmus metody je následující.

Pro všechny pixely obrazu:

- Inicializace $C_{OUT} = 0$
- Je-li $C_{IN} > T$,
  pak $C_{OUT} += 1$

Obrázek 3.3: Příklad převodu obrázku o 256 stupních šedé na obrázek černobílý pomocí prahování.

3.5 Distribuce chyby

Další v pořadí metod **ditheringu** je metoda distribuce chyby. Je založena na prahování, ale s tím rozdílem, že reaguje na chybu vzniklou převodem pixelu na černý nebo bílý. Obraz je třeba procházet po jednotlivých řádcích ve stejném pořadí (v předchozích metodách nebylo třeba dodržovat pořadí při zpracovávání pixelů). Je-li nový pixel převeden na černý, pak se vzniklá chyba rovná hodnotě pixelu před převedením. V opačném případě má chyba velikost rozdílu původního pixelu a maximální hodnoty, které může pixel nabývat. Nyní se podle zvolené metody rozloží (přesněji rozptýlí) chyba na sousední pixely, které ještě nebyly zpracovány (tj. následující pixely na stejném řádku a pixely na následujících řádcích). Ty dostanou novou intenzitu, která ve výsledku redukuje vznikající chyby.

3.5.1 Floyd-Steinberg

Jednou z metod pro distribuci chyby je metoda nazvaná Floyd-Steinberg. Chyba je rozložena mezi sousední čtyři pixely jak je znázorněno na obrázku 3.4. Důležité je mít vhodně zvolené koeficienty pro rozptýlení chyby. Nevhodné rozptýlení může mít za následek horší kvalitu výsledného obrazu. Pomocí této metody lze dosáhnout velice kvalitního výsledku. Příklad je na obrázku 3.5. Algoritmus metody je následující.

Pro všechny pixely obrazu:

- Inicializace $C_{OUT} = 0$
- Určení C_{OUT},
  pro $C_{OUT} = 1 \rightarrow E = C_{IN} - C_{MAX}$
  pro $C_{OUT} = 0 \rightarrow E = C_{IN} - 0$
- Distribuce chyby E sousedním pixelům
- Přidělení výsledné C_{OUT} s ohledem na E v paměti chyby

Obrázek 3.4: Ukázka rozložení chyby u metody Floyd-Steinberg.

Obrázek 3.5: Příklad převodu obrázku o 256 stupních šedé na obrázek černobílý pomocí metody distribuce chyby. Konkrétně metoda Floyd-Steinberg.

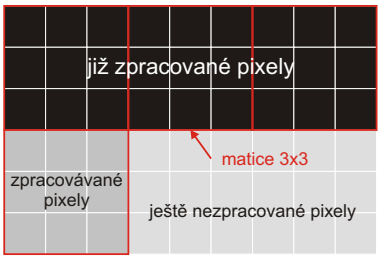
3.6 Maticové rozptýlení

Poslední zmíněnou metodou **ditheringu** je maticové rozptýlení. Základem této metody je vhodně navržená rozptylovací matice. (například 3.3) Rozptylovacích matic existuje celá řada a liší se také například podle toho, zda budou použity pro tisk nebo pro zobrazování na monitoru. Princip této metody spočívá v porovnávání pixelu s hodnotu na určitém

indexu rozptylovací matice. V případě, že pixel má nižší intenzitu, výsledný pixel je černý, v opačném případě bílý. Aby rozptylovací matice pokryla celý obrázek, je nutno ji použít opakovaně. Nejlépe se to představuje jako kachličky nebo dlažba (obrázek 3.6). Příklad metody je na obrázku 3.7. Algoritmus metody je následující.

Pro všechny pixely obrazu:

- Inicializace $C_{OUT} = 0$
- Je-li $C_{IN} > M[x_m, y_m]$,
  pak $C_{OUT} += 1$
  $x_m = x \bmod n$, $y_m = y \bmod n$, n - řád matice

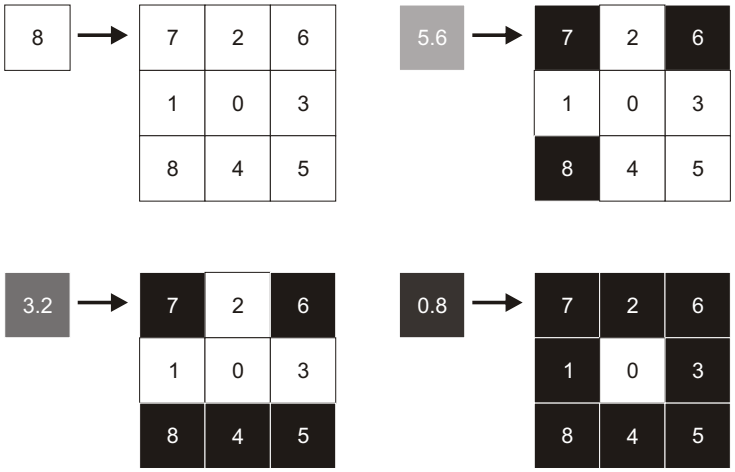


Obrázek 3.6: Ukázka opakovaného použití matice při maticovém rozptýlení (dlažba).

Obrázek 3.7: Příklad převodu obrázku o 256 stupních šedé na obrázek černobílý pomocí maticového rozptýlení.

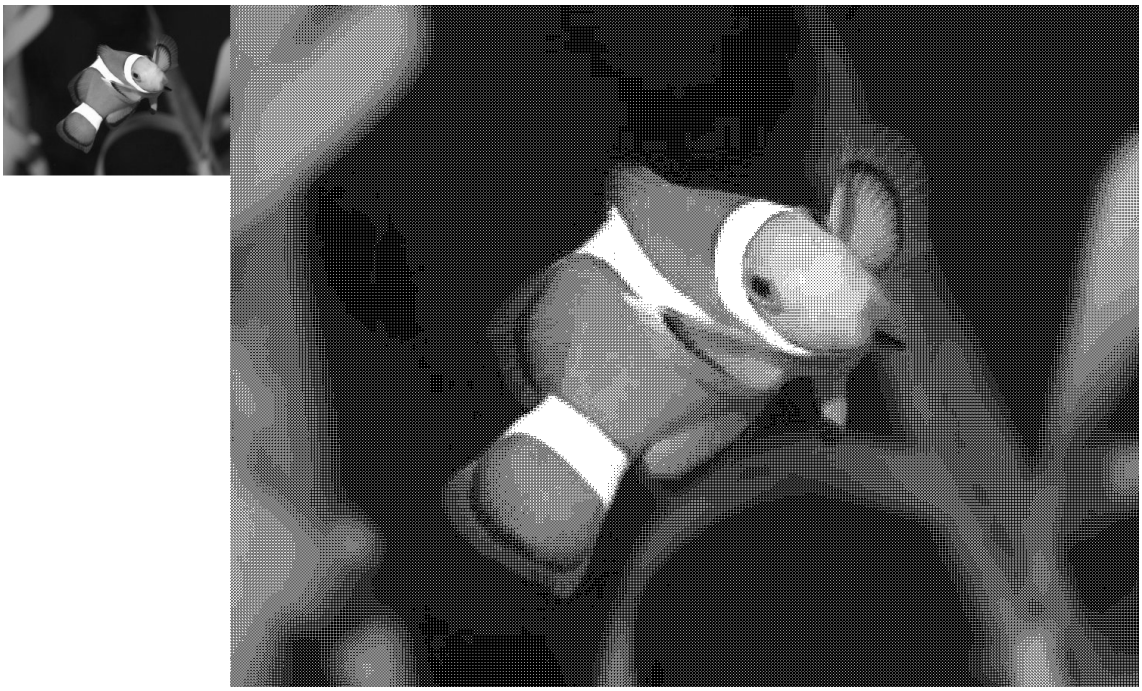
3.7 Maticové polotónování

Tato metoda již nepatří do ditheringu jako metody předchozí, ale řadí se do **halftoningu** (polotónování). Podobně jako u maticového rozptylování je i zde třeba vhodně navržená polotónovací matice (například 3.3). Princip je ovšem odlišný. Každý pixel obrazu je nahrazen celou maticí pixelů. O rozložení pixelů v této matici rozhodují výsledky porovnávání hodnoty zpracovávaného pixelu s hodnotami aktuálních indexů matice. Jde v podstatě o prahování, ovšem s tím rozdílem, že práh je vždy nastaven podle hodnoty na aktuální pozici v polotónovací matici. Rozlišení výsledného obrazu se tedy zvětší v závislosti na řádu matice. Na obrázku 3.8 je pěkně vidět jak je pixel s určitou intenzitou nahrazen maticí nových pixelů. Intenzita pixelu v obrázku je v rozmezí $< 0, 8 >$. Příklad metody je na obrázku 3.9.



Obrázek 3.8: Příklady nahrazení pixelů polotónovacími maticemi o rozměrech 3×3.

$$M = \begin{pmatrix} 160 & 96 & 144 & 80 \\ 32 & 224 & 16 & 208 \\ 128 & 64 & 176 & 112 \\ 0 & 192 & 48 & 240 \end{pmatrix} \tag{3.3}$$



Obrázek 3.9: Příklad převodu obrázku o 256 stupních šedé na obrázek černobílý pomocí maticového polotónování.

Kapitola 4

Požadavky na výukový program

Následující podkapitoly obsahují souhrn požadavků na výukový program.

4.1 Snadná ovladatelnost

Cílová skupina uživatelů, ať už studenti či učitelé, jistě uvítají snadné používání programu. První požadavek tedy je, aby aplikace měla přívětivé grafické uživatelské prostředí. Zdůrazňuji grafické uživatelské prostředí, protože zadávání příkazů například přes příkazový řádek by nebylo příliš efektivní a řada uživatelů by mohla mít s ovládáním problémy. S tím také souvisí do jaké míry je ovládání programu intuitivní. Jednotlivé prvky by měli fungovat tak, jak je uživatel zvyklý. Celkový vzhled aplikace a rozmístění ovládacích prvků bude podobné jako u většiny dnes běžně používaných programů. V případě nutnosti studování složitých manuálů by výukový program ztratil svůj smysl.

4.2 Teorie doplněná praxí

K dobrému pochopení vysvětlované látky je vždy dobré uvést nějaký příklad. Tento fakt je hlavním pilířem výukového programu. Nedílnou součástí tedy musí být popis teoretické části doplněné o praktickou ukázku. Nejlépe se tato skutečnost realizuje pomocí stručného a jednoduchého návodu. V něm budou uvedeny následující informace:

1. Teoretické vysvětlení dané problematiky. V mém případě teoreticky popsána metoda redukce barevného prostoru.
2. Ukázkový obrázek, na kterém je dobře vidět rozdíl mezi obrázkem před a po aplikaci dané redukční metody.
3. Návod, jak si v programu tuto metodu vyzkoušet.
4. Uvedení algoritmu metody, aby bylo patrné, jak doopravdy pracuje.
5. Další doplňující informace.

Kapitola 5

Analýza a objektový návrh aplikace

5.1 Vývojové prostředky a cílová platforma

Program je psán v objektově orientovaném jazyce C++. Tento jazyk ovšem není čistě objektový, hovoříme zde o hybridním objektově orientovaném jazyce (blíže například v [2]). Je také částečně zpětně kompatibilní s procedurálním jazykem C, z kterého ho Bjarne Stroustrup jistými změnami a rozšířeními vyvinul.

Pro tvorbu aplikace, která bude mít přívětivé grafické uživatelské rozhraní (*GUI*) je třeba použít nějaký *toolkit*. Tedy soubor knihoven umožňujících vytvořit GUI na požadované platformě. Vhodné *API* (rozhraní pro programování aplikací) pro C++ poskytuje toolkit **wxWidgets** (zdrojové soubory jsou přístupné na internetu [6]). Tyto knihovny jsou vyvíjeny již od roku 1992 kde hlavním iniciátorem vývoje byl Julian Smart. Velikou výhodou tohoto toolkitu je jeho platformová nezávislost. Je možné ho tedy používat pod různými operačními systémy. Další podrobnosti v [1].

Ke snadnějšímu editování GUI a psaní zdrojového kódu pomocí wxWidgets vytvořil Julian Smart také program **DialogBlocks**. V tomto vývojovém prostředí je tvorba GUI snazší, protože DialogBlocks umožňuje editaci jednotlivých prvků (např. dialogů) s grafickým náhledem. Dokáže vygenerovat také Makefile a jiné soubory potřebné pro překlad.

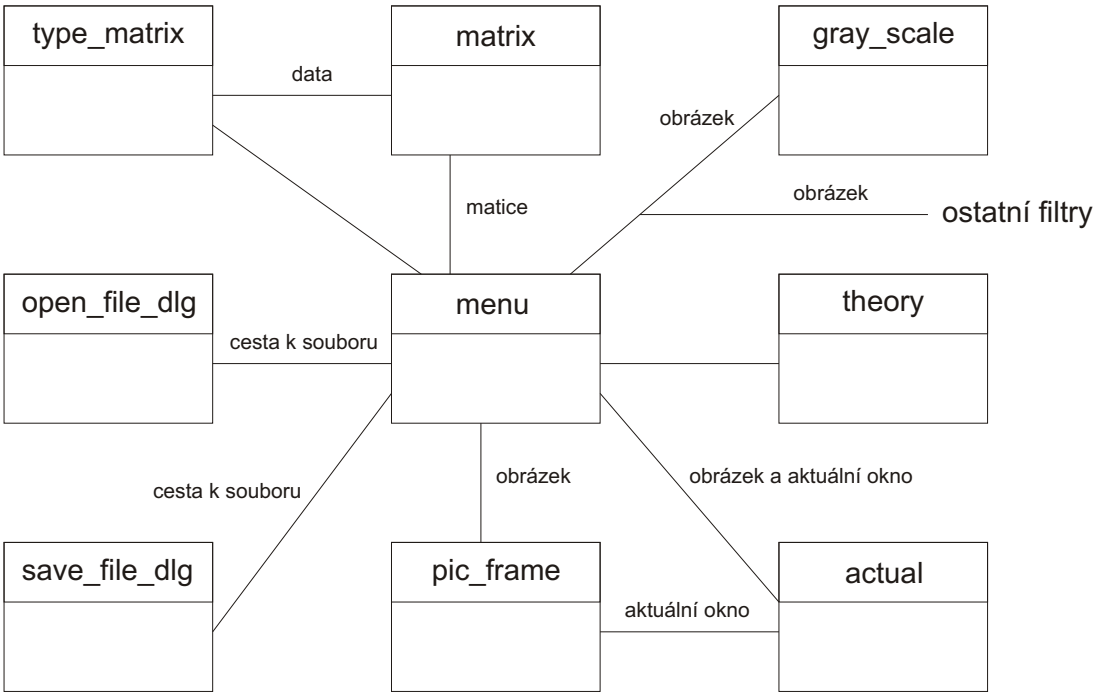
Jako operační systém, ve kterém byla aplikace vyvíjena, byl zvolen *Microsoft Windows XP*. K překladu zdrojových kódů byl použit překladač *VC++, verze 7.0* (součást programu Microsoft Visual Studio 2003).

5.2 Objektový návrh aplikace

Před samotným psaním zdrojových kódu je třeba provést analýzu problému a naplánovat postup řešení. Jak jsem již zmínil v předchozí kapitole, tak program byl vyvíjen v objektově orientovaném jazyce C++. Proto jsem tedy sestavil objektový návrh celé aplikace. Obrázek 5.1 je pouze orientační a jednotlivé části jsou podrobněji popsány v následujících kapitolách.

5.2.1 Stručná charakteristika programu

Ze specifikace zadaní (Výukový program pro demonstraci metod redukce barevného prostoru obrazů) hned vyplývají základní vlastnosti programu. Požadavky na výukový program



Obrázek 5.1: Zjednodušený objektový návrh aplikace.

jsem uvedl již dříve, tak tuhle část zadání nebudu již více rozebírat. Další částí zadání je demonstrace metod redukce barevného prostoru obrazů. Z této věty je jasné, že program obsahuje nějaký mechanizmus zobrazení obrázku a následnou úpravu jeho barev. Pro větší univerzálnost použití je samozřejmě možné načíst libovolný obrázek uložený v podporovaném grafickém formátu. Na něm je demonstrována zvolená redukční metoda. Některé metody je navíc možné upravit nebo nastavit podle požadavků uživatele a až poté aplikovat na obrázek. Funkčnost programu je také rozšířena například o možnost ukládání obrázků nebo zvětšování pomocí lupy.

5.2.2 Třída `menu`

Základní třídou celého programu je třída `menu`. Obsahuje metody pro vytvoření hlavního okna a také zajišťuje správný chod celé aplikace. Objekty ostatních tříd jsou vytvářeny právě v metodách této hlavní třídy.

5.2.3 Třída `theory`

Třída `theory` slouží ke zobrazení teoretické části výukového programu. Její metody dokáží otevřít externí soubor a jeho obsah zobrazit uživateli. K přehlednému formátování textu jsem zvolil použití HTML prohlížeče. HTML kód také podporuje vkládání obrázků, což teoretickou část udělá názornější a přehlednější.

5.2.4 Třídy `open_file_dlg` a `save_file_dlg`

Obě tyto třídy mají velmi podobnou funkčnost. `Open_file_dlg` slouží pro otevření obrázku a `save_file_dlg` pro uložení obrázku. Metody těchto tříd tedy dokáží zobrazit standardní dialog pro listování na disku (popřípadě jiných médiích) a do hlavní třídy `menu` předávají cestu k souboru. `Menu` již zajistí otevření, popřípadě uložení obrázku s danou cestou.

5.2.5 Třída `pic_frame`

Zobrazení obrázku obstarává třída `pic_frame`. `Menu` jí předá obrázek a `pic_frame` se postará o zobrazení do okna. Obsahuje také metody pro přiblížení či oddálení obrázku.

Vzhledem k praktičnosti programu jsem uživateli umožnil současné otevření několika obrázků. Díky tomu bude možné vizuálně lépe porovnávat výsledky jednotlivých metod redukce. Proto je každý obrázek otevřen ve svém vlastním okně. Třída `menu` tedy může vytvořit tolik objektů třídy `pic_frame`, kolik si uživatel bude přát.

5.2.6 Třída `actual`

Existence třídy `actual` je důsledkem vícenásobného otevření různých obrázků. Metody této třídy slouží k uchovávání informací o otevřených obrázcích. Například v jakém okně je který obrázek nebo také které okno bylo naposledy aktivní. Proto třída `actual` komunikuje kromě `menu` i s `pic_frame`. V případě, že uživatel změní aktivní okno s obrázkem na jiné (pokud je oken otevřeno více), `pic_frame` musí o této změně informovat `actual`.

5.2.7 Třídy jednotlivých metod redukce

V obrázku 5.1 je z důvodu nedostatku místa znázorněna jen třída `gray_scale` (převod na stupně šedi). Stejným způsobem jako `gray_scale` komunikují také ostatní třídy pro redukci barev. Pro přehlednost vypíši všechny metody redukce, které jsou implementovány:

1. `gray_scale` (převod na stupně šedé)
2. `random_dither` (náhodné rozptylování)
3. `threshold` (prahování)
4. `floyd_steinberg` (distribuce chyby)
5. `burkes` (také distribuce chyby)
6. `matrix_dither` (maticové rozptýlení)
7. `matrix_half` (maticové polotónování)

Metody těchto tříd aplikují danou metodu redukce na obrázek a vrátí ho zpět hlavní třídě `menu`.

5.2.8 Třída `matrix`

Třída `matrix` obsahuje databázi rozptylovacích nebo polotónovacích matic.

5.2.9 Třída `type_matrix`

Poslední zmiňovaná třída slouží k zobrazení dialogu pro zadávání vlastních hodnot do matic. Komunikuje také s již zmiňovanou třídou `matrix`, do které předá zadaná data.

Kapitola 6

Implementace

Popisování implementace celého programu by bylo velice zdlouhavé a nezajímavé. Proto se v následujících podkapitolách zaměřím spíše na popis implementace jen některých důležitých či zajímavých částí. Při programování jsem postupoval přesně podle objektového návrhu 5.1. Také jména tříd zůstala pochopitelně nezměněna.

6.1 `Menu`, `pic_frame` a ostatní standardní dialogy

Všechna základní okna či dialogy jsou vytvořené děděním ze standardních tříd knihovny wxWidgets. Třídy `menu` a `pic_frame` jsou děděny z třídy pro tvorbu oken `wxFrame`. Dialog pro zadávání vlastních matic `type_matrix` a HTML okno pro zobrazení teorie a nápovědy jsou děděny z třídy `wxDialog`. Pro upřesnění ještě uvedu, že dialogy pro zadávání vlastních hodnot do matice jsou ve skutečnosti tři. Každý je vytvořen pro matice různých rozměrů. Pro matici 2×2 nese jméno `type_matrix_2`, pro matici 3×3 se jmenuje `type_matrix_3` a 4×4 je `type_matrix_4`.

6.1.1 Řídící třída `menu`

U implementace této třídy se ještě zdržím. Jak jsem již naznačil v kapitole o objektovém návrhu třída `menu` řídí chod celé aplikace. Tabulka událostí této třídy obsahuje mnoho reakcí na nejrůznější události. Většinou jsou to reakce na zmáčknutí tlačítka či zvolení nějaké položky roletového menu. Většina obslužných funkcí událostí má následující charakter. Uvedu příklad pro obslužnou funkci tlačítka pro převod obrázku na stupně šedé:

1. Nejprve funkce volá metody třídy `actual` pro zjištění aktuálního okna s obrázkem. To proto, aby se algoritmus redukce provedl na požadovaný obrázek (pokud je otevřených více oken).

2. Následuje zjištění, zda ukazatel na obrázek je platný. V případě, že není otevřený žádný obrázek, nelze aplikovat redukční metodu. V této situaci je uživateli zobrazen standardní dialog s varováním (`wxMessageBox`).

3. Další ošetření se týká zjištění úrovně použití lupy. Pokud je obrázek před použitím redukce zvětšený, musí zůstat zvětšený i po dokončení redukčního algoritmu.

4. Nyní je již vše připraveno k vytvoření objektu redukční třídy. V mém případě například `gray_scale`. Aplikací metody `reduce()` na obrázek je proveden převod barevného obrázku na stupně šedé.

5. Ostatní příkazy se týkají aktualizování ukazatelů či překreslení požadovaného okna.

6.2 Princip použití více redukčních metod na jeden obrázek

Následující odstavec se zabývá problematikou použití více metod redukce (dále jen filtrů) na jediný obrázek. V takovém případě by uživatel musel po aplikaci každého filtru znovu otevírat stejný obrázek. To je nepříjemně zdlouhavé, zvlášť když umístění obrázku na disku je hluboko zanořené ve složkách. Tento problém jsem chtěl prvotně řešit použitím funkčních tlačítek zpět a vpřed. Tím bych docílil jisté výhody, že po aplikování filtru by bylo možné tuto operaci vrátit zpět a použít filtr jiný. Konečné řešení jsem ale nakonec vymyslel ještě jiné. Při otevření nového obrázku je uložen jeho ukazatel jako tzv. originál. V případě použití nějakého redukovacího algoritmu je tento originál zkopírován a filtr použit na tuto kopii. Ta je samozřejmě také zobrazena uživateli. Tímto způsobem je možné libovolně přepínat mezi jednotlivými filtry a znovu a znovu je aplikovat na tentýž obrázek.

6.3 Databáze aktuálních oken a obrázků

O problému vícenásobného otevření obrázku jsem se v předchozím textu již několikrát zmínil. Databázi, která uchovává informace o otevřených oknech a o obrázcích v nich, jsem řešil jako seznam. Jako prvek seznamu je třída `list_item`, která obsahuje členské proměnné:

1. ukazatel na okno,
2. ukazatel na originální obrázek,
3. ukazatel na obrázek,
4. míru použité lupy

Objekt třídy `actual` je deklarován jako globální v třídě `menu`. V ostatních třídách kde se k němu také přistupuje je pro deklaraci použit specifikátor `extern`. Tím je zaručeno že po dobu běhu aplikace je jen jeden objekt třídy `actual`. V konstruktoru této třídy se také vytváří již zmiňovaný seznam, který uchovává informace o všech otevřených oknech s obrázky.

Práce se seznamem je následující. Na poslední pozici v seznamu je vždy záznam o naposledy aktuálním okně. Obdrží-li `pic_frame` zprávu *okno je aktivní* (`EVT_ACTIVATE`), vyhledá se v seznamu požadovaný záznam a je přesunut na konec. Metody pro zjišťování aktuálního okna či obrázku pak již jen vrátí ukazatele z poslední položky seznamu.

6.4 Aplikace filtru

Poslední popisovanou částí v kapitole o implementaci je způsob filtrování obrázku danou metodou. Opět uvedu příklad pro jeden konkrétní filtr (například `threshold`). Ostatní metody jsou podobné.

Nejprve je třeba si zpřístupnit data obrázku. Obrázek je objekt třídy `wxImage`, a proto můžu použít standardní metodu `GetData()`. Nyní pracuji z obrázkem jako s polem bezznaménkových charů (`unsigned char`). Data jsou v něm uložena po řádcích a každý pixel je reprezentován trojicí Red, Green a Blue. Při metodě tedy procházím pixel po pixelu celý obrázek. Porovnávám hodnotu prahu s hodnotou na aktuálním indexu v poli. Na základě

výsledku porovnání upravím hodnotu tohoto znaku. Jde-li ovšem o černobílou redukci, index přepočítávám takovým způsobem, kdy přeskočím hodnoty Green a Blue. Protože obrázek ve stupních šedé má hodnoty všech tří kanálů shodné. Úprava výsledného pixelu se tedy v tomto případě také vztahuje na všechny tři kanály.

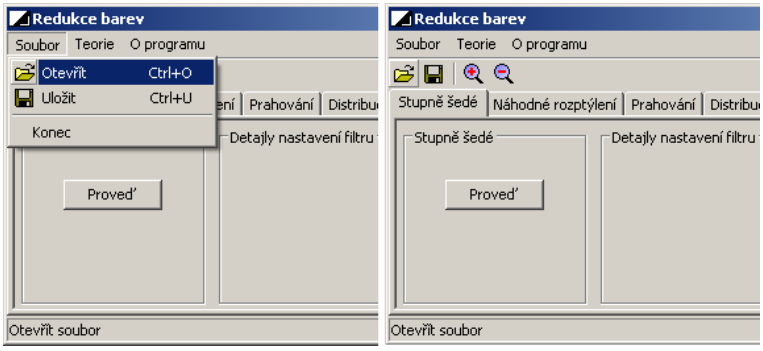
Kapitola 7

Práce s programem

V následujících kapitolách stručně popíši práci s programem. Představím základní použití filtrů, zobrazení nápovědy, ale také upozorním na situace, kde je třeba dávat pozor.

7.1 Načtení obrázku

K zobrazení dialogu pro načtení nového obrázku můžete použít buď položku roletového menu *Soubor* → *Otevřít*, či obrázek žluté složky v menu nástrojů, popřípadě použít klávesovou zkratku *Ctrl+O*. Názorně na obrázku 7.1.

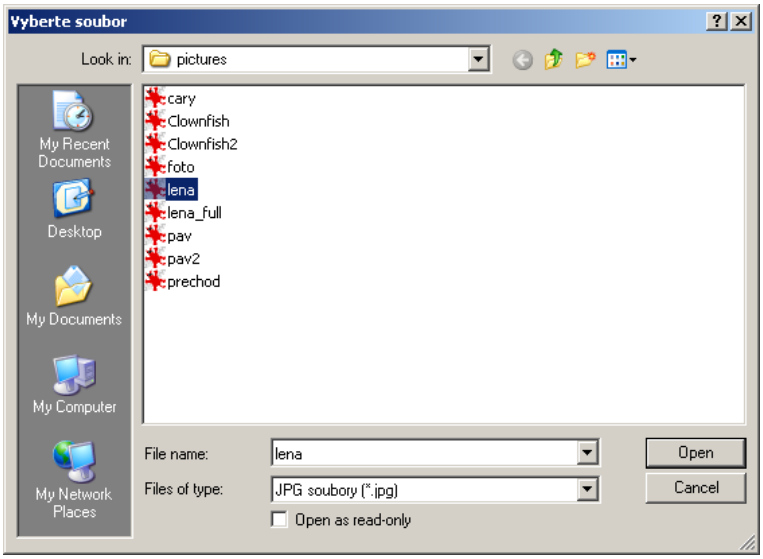


Obrázek 7.1: Otevření nového obrázku.

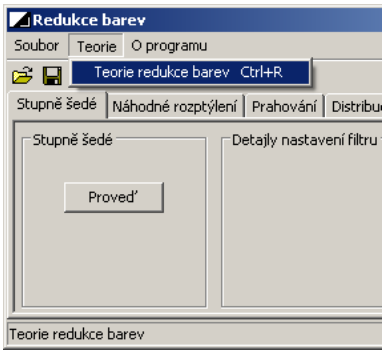
Nyní vyberte požadovaný obrázek a stiskněte *Otevřít* (popřípadě *Open*, záleží na vašem OS). Můžete také použít filtrování souborů podle přípon. Například zvolit pouze JPEG obrázky. Názorně na obrázku 7.2.

7.2 Zobrazení teorie

Pro zobrazení teorie opět slouží roletová nabídka. Zvolte *Teorie* → *Teorie Redukce Barev*. I zde funguje klávesová zkratka *Ctrl+R*. Názorně na obrázku 7.3.



Obrázek 7.2: Dialog pro výběr obrázku s použitím filtrování podle přípony.



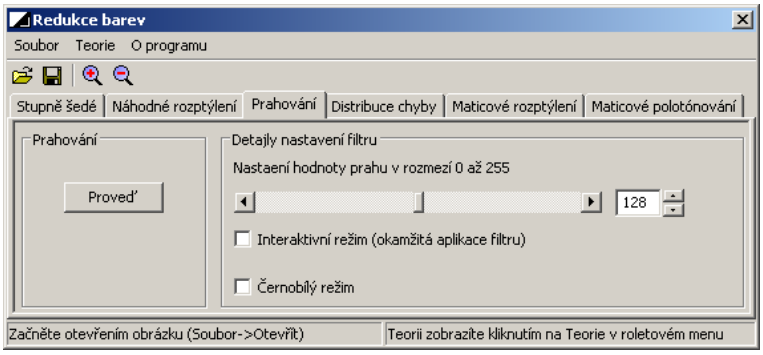
Obrázek 7.3: Zobrazení teorie.

7.3 Záložka prahování

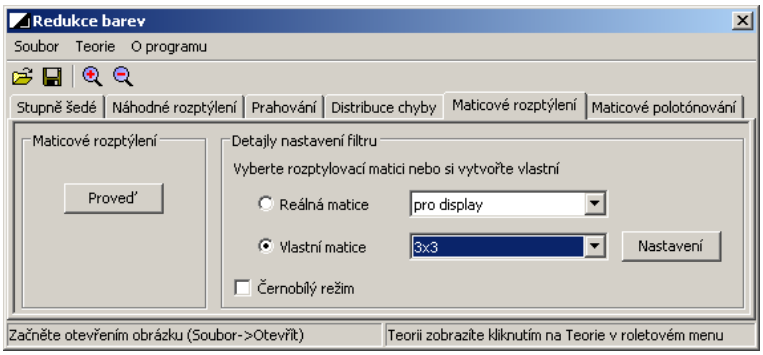
Vzhledem k podobnosti použití popíši jen některé metody redukce. Jako první se pozastavím u metody prahování. Zde může uživatel nastavit hodnotu prahu. Slouží k tomu buď posuvník, nebo textové pole pro zadávání čísel. Rozsah hodnot pro práh je od 0 do 255. Další možností je použití tzv. *Interaktivního režimu*. Funguje tak, že při jakékoli změně prahu (ať už na posuvníku nebo číselně) dojde k okamžitému překreslení obrázku. Poslední možné nastavení je *Černobílý režim*. V případě, že zaškrtnete toto pole, bude na obrázek aplikován černobílý filtr. Názorně na obrázku 7.4.

7.4 Záložka maticové rozptýlení

V nastavení tohoto filtru se nacházejí dva seznamy. Jeden z nich umožňuje výběr již hotové matice, zatímco druhý slouží pro tvorbu vlastní matice. K přepínání těchto dvou módů slouží přepínače *Reálná matice* a *Vlastní matice*. Názorně obrázek 7.5



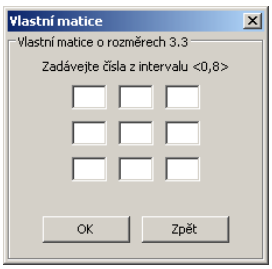
Obrázek 7.4: Záložka metody pro prahování.



Obrázek 7.5: Záložka metody pro maticové rozptýlení.

7.4.1 Nastavení vlastní matice

K zobrazení dialogu, ve kterém je možné nastavit vlastní matici, slouží tlačítko *Nastavení*. Ještě před samotným zmáčknutím tlačítka vyberte ze seznamu rozměry matice. V závislosti na zvolených rozměrech se otevře požadovaný dialog. Do jednotlivých buněk můžete psát libovolné hodnoty, musí ovšem být v požadovaném intervalu. Zadávání ukončíte stiskem tlačítka *OK*. Názorně na obrázku 7.6.

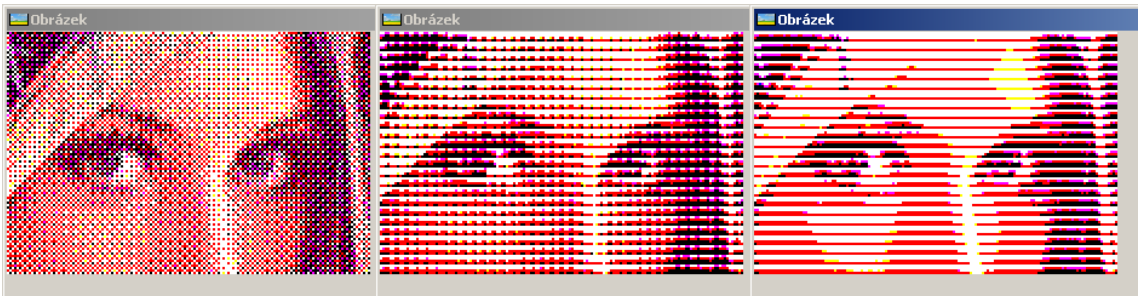


Obrázek 7.6: Zadávání vlastních hodnot pro rozptylovací matici.

7.4.2 Nevhodně zadaná matice

Při nevhodně zvolených hodnotách pro matici můžete dosáhnout poněkud horšího vzhledu obrázku. Můžou se objevit různé pruhy či artefakty v obraze. Pěkně vidět je to na obrázku 7.7, kde byly použity rozptylovací matice 7.1:

$$M1 = \begin{pmatrix} 10 & 6 & 9 & 5 \\ 2 & 14 & 1 & 13 \\ 8 & 4 & 11 & 7 \\ 0 & 12 & 3 & 15 \end{pmatrix} M2 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{pmatrix} M3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 4 & 4 & 4 & 4 \\ 8 & 8 & 8 & 8 \\ 12 & 12 & 12 & 12 \end{pmatrix}$$



Obrázek 7.7: Aplikování rozptylovacích matic (zleva M1, M2, M3).

Kapitola 8

Závěr

Výukový program byl vyvíjen a testován pod operačním systémem Microsoft Windows XP Professional (SP2, version 2002). Vzhledem k použitému multiplatformního toolkitu wxWidgets byl testován i na jiném operačním systému. Konkrétně se jednalo o Kubuntu Linux (s jádrem 2.6.15-27-386). Funkčnost programu zde ovšem byla omezená, proto doporučuji ho používat pouze pod systémem Windows. Zdrojové soubory celého programu včetně programové dokumentace jsou uloženy na přiloženém CD. Kompletní obsah CD je vypsán v dodatku A.

Je třeba upozornit na fakt, že se jedná o výukový program a nikoli o program pro úpravu obrázků. S tím také souvisí několik následující drobností. Podporované formáty načítaných obrázků jsou omezeny na čtyři nejpoužívanější. Jedná se o obrázky ve formátu BMP, JPG, GIF a PNG. Další věcí, na kterou je třeba upozornit, je to, že při ukládání zredukovaného obrázku nedochází k úpravě jeho barevné hloubky. Například i černobílé obrázky mohou být uloženy jako 24 bitové. Pro výukové účely nebylo třeba implementovat tyto rozšířené funkce. Uložit obrázek lze ve formátech BMP, JPG a PNG.

A právě úpravu barevné hloubky by bylo možné implementovat jako možné rozšíření programu. Další rozšíření, které by jistě nalezlo v programu uplatnění, může být zadávání vlastních rozptylovacích či polotónovacích matic neomezených rozměrů. Doposud je možné zadávat pouze matice o rozměrech 2×2, 3×3 nebo 4×4. Poslední rozšíření, které mě napadá, je implementovat možnost postupného spouštění filtrů na jeden obrázek. Doposud je to tak, že po zredukování obrázku nějakou metodou se musí obrázek uložit, potom znovu načíst a pokračovat úpravou třeba pomocí jiného filtru. Ovšem toto rozšíření je dosti spekulativní, protože na výuku nemá téměř žádný vliv. Ostatní vylepšení či změny v programu mohou navrhnout až uživatelé, kteří budou program používat.

Literatura

[1] Bližňák, M.: Knihovna wxWidgets. [online], slajdy k přednáškám na FAI UTB Zlín. URL http://vyuka.fai.utb.cz/mod/resource/index.php?id=17

[2] Eckel, B.: *Knihovna programátora: Myslíme v jazyku C++, 1. díl*. Grada, 2000, ISBN 80-247-9009-2.

[3] Fraser, B.: *Správa barev: Průvodce profesionála v grafice a pre-pressu*. Computer Press, 2004, ISBN 80-722-6943-7.

[4] Sochor, J.: Barva a barevné vidění. [online], 2007, slajdy k přednáškám na FI MU Brno.
URL http://www.fi.muni.cz/~sochor/M4730/Slajdy/

[5] Wikipedia: International Commission on Illumination. [online], [cit. 27.dubna 2007].
URL
http://en.wikipedia.org/wiki/International_Commission_on_Illumination

[6] WxWidgets: Cross-Platform GUI Library. [online], [cit. 9.května 2007].
URL http://www.wxwidgets.org/

[7] Žára, J.: *Moderní počítačová grafika*. Computer Press, 2004, ISBN 80-251-0454-0.

Dodatek A

Obsah přiloženého CD

Na přiloženém CD se nacházejí následující soubory:

- složka `technická zpráva`:
  - `bp.pdf` – technická zpráva bakalářské práce
  - `zdrojové soubory` – složka obsahuje všechny zdrojové soubory potřebné pro překlad technické zprávy
- složka `program`:
  - `spustitelný program` – složka obsahuje přeložené zdrojové soubory programu včetně spustitelného souboru `color_reduction.exe`
  - `zdrojové soubory` – složka obsahuje všechny zdrojové soubory programu potřebné pro překlad. Při použití programu `nmake` (překladač VC++) je k dispozici soubor `makefile.vc`
- složka `dokumentace`:
  - `dokumentace.html` – soubor, který spustí titulní stránku programové dokumentace. Ta je vygenerována prostřednictvím programu Doxygen
  - `html` – složka obsahuje zdrojové soubory programové dokumentace
- složka `wxWidgets`:
  - `wxMSW-2.6.3-Setup-1.exe` – instalační soubor pro instalaci knihovny wxWidgets