

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
KIV/KPG

Photoshop

Pavel Zelenka
A16B0176P
zelenkap@students.zcu.cz

27. dubna 2018

1 Zadání

Zadáním úkolu je vytvoření programu umožňujícího aplikaci efektů na načteném rastrovém obrázku.

2 Analýza problému

Aplikace bude mít oproti úloze na cvičení čtyři nové efekty, tj. **negativ**, **sepia**, **mozaika** a **rozmazání**.

2.1 Negativ

Negativ je tonální a barevná inverze obrazu. Je-li uvažováno 8 bitů (tj. 256 tónů) na každý barevný kanál, pro získání negativu lze vzít absolutní hodnotu po odečtení konstanty 255 od každého barevného kanálu.

Získání negativní hodnoty pixelu:

$$color = [red, green, blue]^T$$

$$negative_red = |red - 255|$$

$$negative_green = |green - 255|$$

$$negative_blue = |blue - 255|$$

$$negative_color = [negative_red, negative_green, negative_blue]^T$$

2.2 Sepia

Sepia je jednobarevný hnědotónový obraz. V prvním kroce se obraz převede na černobílý, to se provede sečtením hodnot všech barevných kanálů a vydělením jejich počtem. Hnědého nádechu obrazu lze dosáhnout zvýšením hodnot červeného a zeleného kanálu a snížením hodnoty modrého kanálu. Při manipulaci s tónama je nutné ošetřit, aby hodnoty byly v intervalu od 0 do 255. V případě překročení horní meze se nastaví hodnota na 255, naopak v případě překročení dolní meze se hodnota nastaví na 0.

Získání šedotónové hodnoty pixelu:

$$color = [red, green, blue]^T$$

$$gray_tone = \frac{red + green + blue}{3}$$

$$gray_color = [gray_tone, gray_tone, gray_tone]^T$$

Přidání sepiového efektu:

$$\exists depth \in \langle -255, 255 \rangle \subset \mathbb{Z}$$

$$\exists intensity \in \langle -255, 255 \rangle \subset \mathbb{Z}$$

$$gray_color = [gray_tone_{red}, gray_tone_{green}, gray_tone_{blue}]^T$$

$$sepia_tone_{red} = gray_tone_{red} \cdot 2 \cdot depth$$

$$sepia_tone_{green} = gray_tone_{green} \cdot depth$$

$$sepia_tone_{blue} = gray_tone_{blue} - intensity$$

$$sepia_color = [sepia_tone_{red}, sepia_tone_{green}, sepia_tone_{blue}]^T$$

2.3 Mozaika

Mozaika je obraz tvořený z kostek, počet těchto kostek může být libovolný. Přichází zde v úvahu, aby si uživatel mohl poměr rozdělení obrázku specifikovat zadáním koeficientu. Tímto koeficientem by se vynásobila šířka obrázku a výsledkem by byl počet kostek na šířku. Tato implementace vyžaduje, aby koeficient byl v intervalu od 0 do 1. Jako minimální hodnotu koeficientu ovšem nelze zvolit 0, obrázek by se vždy měl skládat minimálně z jedné kostky. V případě koeficientu blížíci se 1 zase nastává situace, kdy počet kostek odpovídá přibližně počtu pixelů a na obrázku se neprovedou žádné změny.

Určení velikosti kostky mozaiky:

$$\exists precent \in (0, 1) \subset \mathbb{R}$$

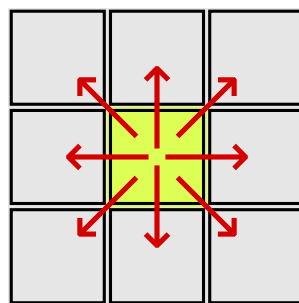
$$\exists image_width \in \mathbb{N}$$

$$square_side = image_width \cdot precent$$

Průchod obrázku se bude provádět vždy po násobcích velikosti čtverce, všechny pixely v tomto čtverci se přebarví na jednotnou barvu. Barvu lze zvolit více způsoby, v aplikaci se budu zabývat jednorůchodovým způsobem, kdy je nejvhodnější zvolit barvu horního levého pixelu a dvouprůchodovým způsobem, kdy se udělá průměr barev všech pixelů ve čtverci.

2.4 Rozmázání

Rozmázání obrazu se provádí průchodem obrazu pixel po pixelu a průměrováním barvy se sousedními pixely. Problém nastává u krajních pixelů, kdy lze průměrování dělat s menším počtem sousedů nebo krajní pixely úplně vynechat.



Obrázek 1: Sousední pixely při zpracovávání obrazu (tzv. 8-connectivity)

Při průměrování barvy lze každému pixelu dát váhu, kterou se jeho hodnota vynásobí nebo vydělí v závislosti na implementaci. V případě násobení by váha měla být v intervalu od 0 do 1 (tzn. bude-li mít každý z 9 pixelů stejnou váhu, tak onou váhou bude hodnota 0.111).

3 Popis řešení

Aplikace je naprogramována v jazyce **Java** s použitím grafických knihoven **JavaFX**. Algoritmy efektů jsou umístěny v samostatné třídě *BasicEffects* v balíčku *Effects*. Mimo nových efektů třída *BasicEffects* obsahuje i efekty, které byly použity na cvičení.

3.1 Negativ

```
1 for každý řádek obrázku do
2   for každý sloupec obrázku do
3     barva = barva pixelu na pozici;
4     červená = | červená složka barvy − 255 |;
5     zelená = | zelená složka barvy − 255 |;
6     modrá = | modrá složka barvy − 255 |;
7     nastav novou barvu pixelu na pozici s upravenými složkami;
8   end
9 end
```

Algorithm 1: Vytvoření efektu negativu obrazu

3.2 Sepia

Input: hloubka, intenzita

```
1 for každý řádek obrázku do
2   for každý sloupec obrázku do
3     barva = barva pixelu na pozici;
4     červená = červená složka barvy;
5     zelená = zelená složka barvy;
6     modrá = modrá složka barvy;
7     šedá = (červená + zelená + modrá) / 3;
8     červená = šedá + 2 · hloubka;
9     zelená = šedá + hloubka;
10    modrá = šedá − intenzita;
11    složky s hodnotou nad 255 nastav na 255;
12    složky s hodnotou pod 0 nastav na 0;
13    nastav novou barvu pixelu na pozici s upravenými složkami;
14  end
15 end
```

Algorithm 2: Vytvoření efektu sepie obrazu

3.3 Mozaika s jedním průchodem

Input: šířka, koeficient

```
1 velikost čtverce = šířka · koeficient;
2 if velikost čtverce > 0 then
3   | velikost čtverce = šířka / velikost čtverce;
4 else
5   | velikost čtverce = šířka
6 end
7 for každý řádek obrázku, který je násobkem velikosti čtverce do
8   | for každý sloupec obrázku, který je násobkem velikosti čtverce do
9     | barva = barva pixelu na pozici;
10    | for každý řádek čtverce do
11      | for každý sloupec čtverce do
12        | if procházený bod je platným bodem obrázku then
13          | nastav barvu pixelu;
14        | end
15      | end
16    | end
17  | end
18 end
```

Algorithm 3: Vytvoření efektu mozaika obrazu s jedním průchodem čtverce

3.4 Mozaika s dvěma průchody

Input: šířka, koeficient

```
1 velikost čtverce = šířka · koeficient;
2 if velikost čtverce > 0 then
3   | velikost čtverce = šířka / velikost čtverce;
4 else
5   | velikost čtverce = šířka
6 end
7 for každý řádek obrázku, který je násobkem velikosti čtverce do
8   | for každý sloupec obrázku, který je násobkem velikosti čtverce do
9     | počet pixelů = 0;
10    | červená = 0;
11    | zelená = 0;
12    | modrá = 0;
13    | for každý řádek čtverce do
14      | for každý sloupec čtverce do
15        | if procházený bod je platným bodem obrázku then
16          | červená = červená + červená složka procházeného pixelu;
17          | zelená = zelená + zelená složka procházeného pixelu;
18          | modrá = modrá + modrá složka procházeného pixelu;
19          | počet pixelů = počet pixelů + 1;
20        | end
21      | end
22    | end
23    | červená = červená / počet pixelů;
24    | zelená = zelená / počet pixelů;
25    | modrá = modrá / počet pixelů;
26    | složky s hodnotou nad 255 nastav na 255;
27    | složky s hodnotou pod 0 nastav na 0;
28    | vytvoř novou barvu se složkami červená, zelená, modrá;
29    | for každý řádek čtverce do
30      | for každý sloupec čtverce do
31        | if procházený bod je platným bodem obrázku then
32          | nastav novou barvu pixelu na procházené pozici;
33        | end
34      | end
35    | end
36  | end
37 end
```

Algorithm 4: Vytvoření efektu mozaika obrazu s dvěma průchody čtverce

3.5 Rozmazání

Input: pole vah pixelů

```
1 for každý řádek obrázku mimo okrajových do
2   for každý sloupec obrázku mimo okrajových do
3     index váhy = 0;
4     červená = 0;
5     zelená = 0;
6     modrá = 0;
7     for řádek - 1, aktuální řádek, řádek + 1 do
8       for sloupec - 1, aktuální sloupec, sloupec + 1 do
9         if procházený bod je platným bodem obrázku then
10          červená = červená + červená složka pixelu / váha pixelu;
11          zelená = zelená + zelená složka pixelu / váha pixelu;
12          modrá = modrá + modrá složka pixelu / váha pixelu;
13          index váhy = index váhy + 1;
14         end
15       end
16     end
17     složky s hodnotou nad 255 nastav na 255;
18     složky s hodnotou pod 0 nastav na 0;
19     nastav novou barvu pixelu na pozici s upravenými složkami;
20   end
21 end
```

Algorithm 5: Vytvoření efektu rozmazání obrazu

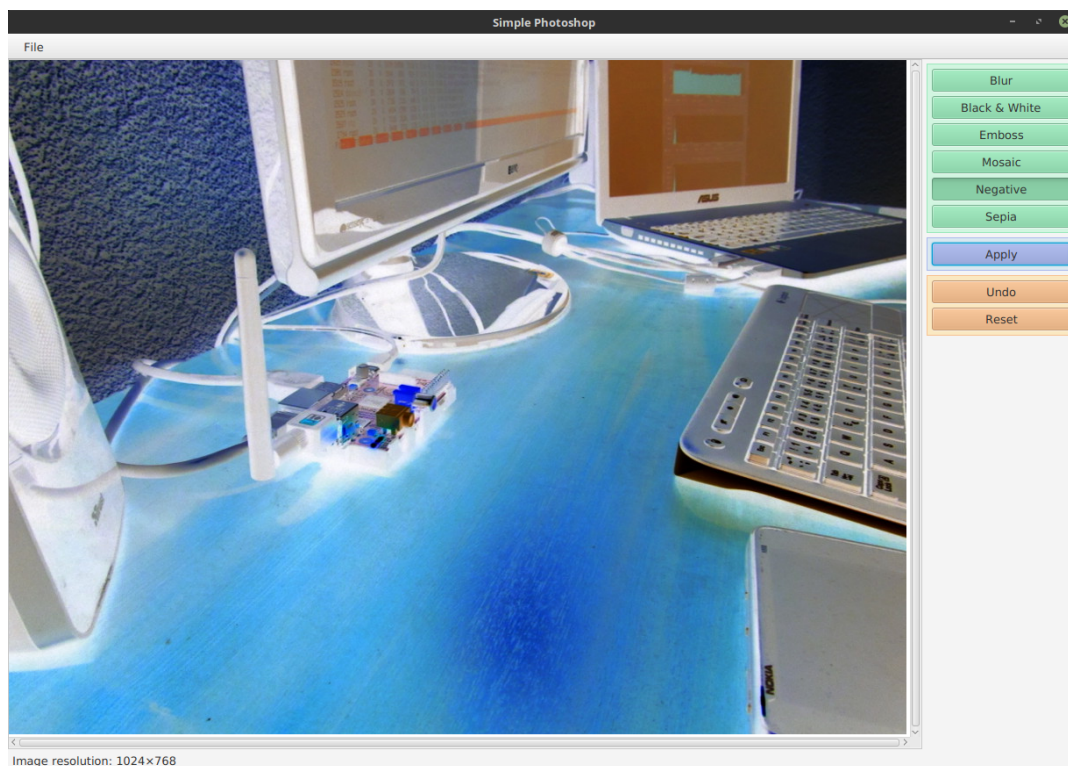
4 Uživatelská dokumentace

Aplikace byla testována na operačním systému **GNU/Linux** s nainstalovaným **Java Development Kit** ve verzi 1.8.0_162. Spouštěcí soubor aplikace **SimplePhotoshop.jar** se nachází ve složce *App*.

Po spuštění aplikace se zobrazí okno s výchozím obrázkem. V pravém panelu se nachází tlačítka efektů **Blur** pro rozmazání, **Black & White** pro černobílý obraz, **Emboss** pro tepaný obraz, **Mosaic** pro mozaiku, **Negative** pro negativ a **Sepia** pro efekt sepie. Pro použití efektu je vždy nutné kliknout na potvrzovací tlačítko **Apply**. Obraz lze vždy vrátit o jeden krok zpět tlačítkem **Undo**. Obnovit původní obrázek lze tlačítkem **Reset**.

Skrze tlačítko **Open** v nabídce **File** lze načíst obrázky ve formátu PNG, JPG a BMP. Ve stejné nabídce skrze tlačítko **Save As...** lze obrázek uložit ve formátu PNG.

Ve spodním panelu je zobrazeno rozlišení načteného obrázku. V případě použití efektu se zobrazí v panelu i čas běhu algoritmu.



Obrázek 2: Okno aplikace

5 Závěr

Aplikaci jsem testoval na obrázku `lena.bmp`, tedy na shodném obrázku jako byl použit na cvičení. Aplikaci efektů na obrázku o rozměrech 512×512 pixelů jsem testoval na notebooku s procesorem Intel Core i7-3610QM a na operačním systému GNU/Linux.

5.1 Výsledky pro negativ



Obrázek 3: Efekt negativ na výchozím obrázku

počet pokusů:	5
nejlepší čas:	31 ms
nejhorší čas:	48 ms
průměrný čas:	36 ms

5.2 Výsledky pro sepia



Obrázek 4: Efekt sepia na výchozím obrázku

počet pokusů:	5
nejlepší čas:	27 ms
nejhorší čas:	34 ms
průměrný čas:	31 ms

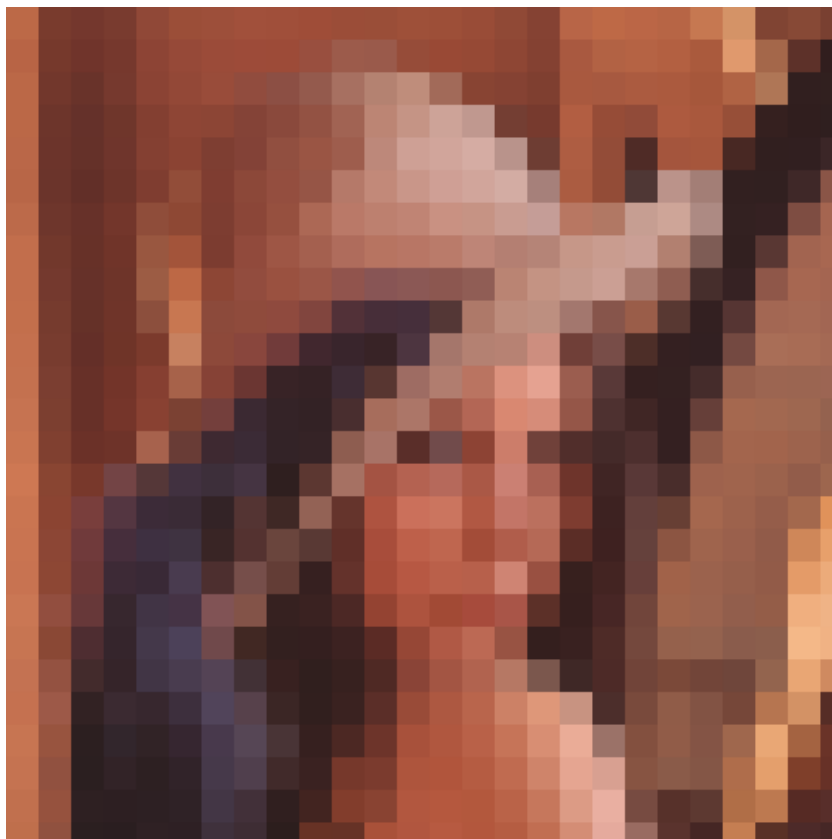
5.3 Výsledky pro mozaiku s jedním průchodem



Obrázek 5: Efekt mazaika s jedním průchodem na výchozím obrázku

počet pokusů:	5
nejlepší čas:	7 ms
nejhorší čas:	16 ms
průměrný čas:	13 ms

5.4 Výsledky pro mozaiku s dvěma průchody



Obrázek 6: Efekt mazaika s dvěma průchody na výchozím obrázku

počet pokusů:	5
nejlepší čas:	16 ms
nejhorší čas:	29 ms
průměrný čas:	22 ms

5.5 Výsledky pro rozmazání



Obrázek 7: Efekt rozmazání na výchozím obrázku

počet pokusů:	5
nejlepší čas:	126 ms
nejhorší čas:	134 ms
průměrný čas:	128 ms

6 Reference

Blurring for Beginners – JH Labs. [online]. Dostupné z: www.jhlabs.com/ip/blurring.html