PAINT++

Grafický editor s obrazovými filtry

Programování a databáze

David Dvořák, Daniel Malík, Ondřej Kutlvašr, Jaroslav Dvořáček

V4-PROG1

OBSAH

https://github.com/paintpp/

1. Úvod

Paint++ je desktopová aplikace pro úpravu obrázků v Javě s využitím JavaFX. Aplikace obsahuje několik filtrů, které je možno aplikovat na nahraný obrázek. Filtrů lze aplikovat více zaráz a fotku je následně možné uložit.

1.1. Hlavní funkce aplikace

V aplikace lze provádět následné operace:

- Načítat a ukládat obrázky ve formátech JPG a PNG
- Generovat obrázky
- Aplikovat 14 různých obrazových filtrů s možností konfigurace
- Upravovat obrázky se zachováním originálu a možností porovnání obou stavů
- Přibližovat a posouvat obrázek

2. Závislosti

Základní technologie a Build

- Java 25: Programovací jazyk
- Maven: Build systém pro správu projektu

Uživatelské rozhraní (GUI Framework)

- JavaFX (verze 25): Hlavní GUI framework
    - javafx-controls: Základní sada GUI komponent
    - javafx-swing: Modul pro integraci se Swingem (využíváno pro ukládání)

Vzhled a stylování

- AtlantaFX (verze 2.1.0): Knihovna pro moderní CSS styly v JavaFX
    - atlantafx-base: Základní modul stylů

Ikony

- Ikonli (verze 12.4.0): Sada Material Design ikon
    - ikonli-javafx: Podpora ikon pro JavaFX
    - ikonli-materialdesign2: Konkrétní balíček Material Design ikon

Nástroje a testování

- DevToolsFX (verze 1.0.1): Vývojářské nástroje (devtoolsfx-gui)
- JUnit Jupiter (verze 5.12.1): Knihovna pro Unit testování

3. Architektura aplikace

Při vývoji se uplatnil vzor MVC, který důsledně odděluje logiku od uživatelského rozhraní. Díky modulární architektuře lze do projektu jednoduše implementovat nové filtry.

3.1 Vrstvy aplikace

1. **Prezentační vrstva (UI)** – JavaFX komponenty pro zobrazení a interakci s uživatelem
2. **Logická vrstva** – správa obrázků, filtrů a jejich aplikace
3. **Datová vrstva** – práce s pixelovými buffery a I/O operace

3.2 Klíčové komponenty

ImageManager – správce obrázků využívající reaktivní vzor Observer. Uchovává aktuální zdroj obrázku a stav zobrazení.

FilterManager – správce filtrů s podporou asynchronního zpracování. Automaticky reaguje na změny v seznamu filtrů a aplikuje je na pozadí.

BufferBackedImage – wrapper nad JavaFX Image poskytující přímý přístup k pixelovému bufferu pro efektivní manipulaci s daty.

4. Struktura projektu

Zdrojový kód je organizován do logicky členěných balíčků pod kořenovým balíčkem com.purkynka.paintpp.

4.1. Přehled balíčků

Balíček	Popis
logic	Aplikační logika a zpracování dat
→ logic.filter	Správa a implementace filtrů
→ logic.filter.imagefilter	Pixelové filtry
→ logic.filter.kernelfilter	Konvoluční filtry
→ logic.image	Správa obrázků a I/O operace
→ logic.observable	Reaktivní datové struktury
→ logic.util	Pomocné utility
ui	Uživatelské rozhraní
→ ui.element	UI komponenty
→ ui.stage	Okna a popup dialogy

4.2. Struktura souborů

5. Správa obrázků

5.1. BufferBackedImage

Třída **BufferBackedImage** obaluje JavaFX obrázek a poskytuje efektivní přístup k obrazovým datům pomocí IntBuffer. Umožňuje přímou manipulaci s pixely bez nutnosti konverze.

Hlavní metody:

- **getPixelIntBuffer()** – získá přístup k raw pixel datům
- **getPixelBuffer()** – získá PixelBuffer pro JavaFX rendering
- **getImage()** – získá JavaFX Image pro zobrazení
- **getImageSize()** – získá rozměry obrázku

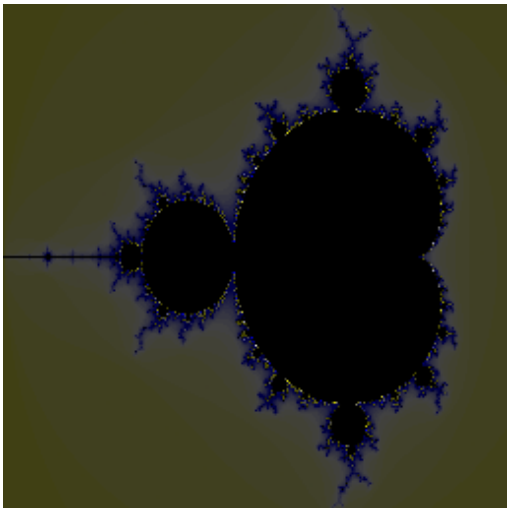
5.2. ImageProvider

Abstraktní třída definující zdroj obrázku. Existují dvě implementace:

- **LoadedImageProvider** – obrázek načtený ze souboru (JPG, PNG)
- **GeneratedImageProvider** – procedurálně generovaný obrázek

5.3. Typy generovaných obrázků

Typ	Popis
X / Y / XY	Gradient založený na pozici pixelu
XY_AVERAGE	X a Y s průměrem v modrém kanálu
SIN	Sinusové vzory z pozic
MANDELBROT	Mandelbrotova množina (fraktál, pomalejší)



Mandelbrot

Gradient

5.4. ImageIO

Třída pro načítání a ukládání obrázků:

- **openImageURI()** – otevře dialog pro výběr souboru a vrátit jeho URI
- **saveImage(Image)** – otevře dialog pro uložení a exportování obrázku

Podporované formáty: JPG, JPEG, PNG

6. Filtry

Systém filtrů je jádrem aplikace. Filtry jsou rozděleny do dvou kategorií podle způsobu zpracování obrazových dat.

6.1. Hiararchie filtrovacích tříd

Všechny filtry dědí z abstraktní třídy ImageFilter, která poskytuje:

- Systém kešování výsledků pro optimalizaci výkonu
- Měření času výpočtu
- Mechanismus dirty flagů pro invalidaci keše
- Abstraktní metodu modifyPixelBuffer() pro implementaci filtru

Konvoluční filtry navíc dědí z třídy KernelFilter, která přidává:

- Automatickou normalizaci konvolučního jádra
- Abstraktní metodu constructKernel() pro definici matice
- Implementaci aplikace konvoluce na všechny pixely

6.2. Pixelové filtry

Pixelové filtry jsou filtry zpracovávající každý pixel nezávisle na okolních pixelech.

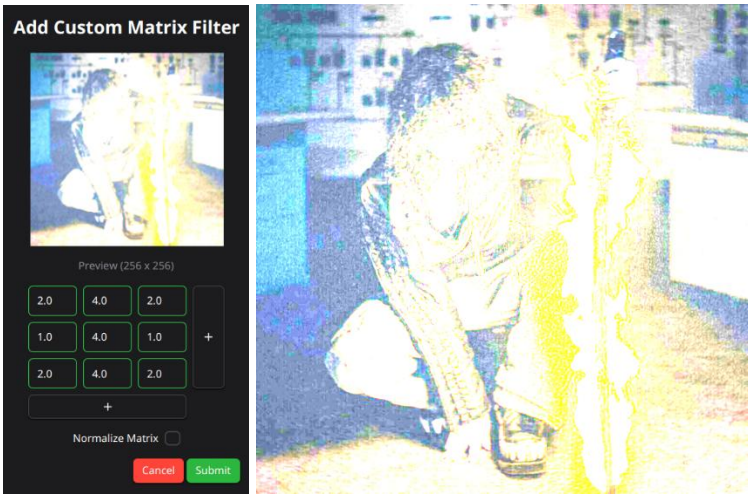
Filtr	Funkce	Vstup / Parametry
Black & White	Převede obrázek na odstíny šedi	*Bez parametrů*
Negative	Invertuje barvy obrázku	*Bez parametrů*
Threshold	Aplikuje prahování s nastavenou hodnotou	Hodnota prahu (0-255)
Pixelize	Sloučí pixely do větších bloků	Velikost bloku (px)
Noise	Přidá šum do obrázku	Intenzita šumu (%)
HSB	Upraví odstín, sytost a jas	3 posuvníky (Hue, Sat, Bright)
Colorizer	Přidá nádech zvolené barvy	Výběr barvy
Color Temperature	Změní teplotu barev	Teplota

Originální obrázek *Obrázek po aplikaci filtru Colorizer*

6.3. Filtry využívající konvoluční matice

Filtry využívající konvoluční matice – je spočítána nová hodnota pixelu z jeho okolí.

Filtr	Funkce	Vstup / Parametry
Gaussian Blur	Rozmaže obrázek pomocí Gaussovy funkce	Intenzita (sigma)
Sharpen	Doostří hrany v obrázku	*Bez parametrů*
Emboss	Vytvoří efekt reliéfu	*Bez parametrů*
Horizontal Edge Detect	Detekuje horizontální hrany (Sobel)	*Bez parametrů*
Vertical Edge Detect	Detekuje vertikální hrany (Sobel)	*Bez parametrů*
Custom Matrix	Aplikuje uživatelovu konvoluční matici	Matice



Přidání filtru s vlastní maticí

6.4. Princip fungování

Kešování

Při prvním použití filtru se výsledek uloží do paměti. Při opakovaném použití (bez změny parametrů) se pouze zkopíruje kešovaný výsledek – to je mnohem rychlejší než přepočítávat celý obrázek.

Dirty flag

Pokud se změní parametry filtru, označí se jako "dirty". Při dalším použití se přepočítá a uloží nová verze do keše.

Měření času

Každý filtr měří dva časy – calculationTime (skutečný čas výpočtu) a copyTime (čas kopírování z keše).

6.5. Příklad implementace

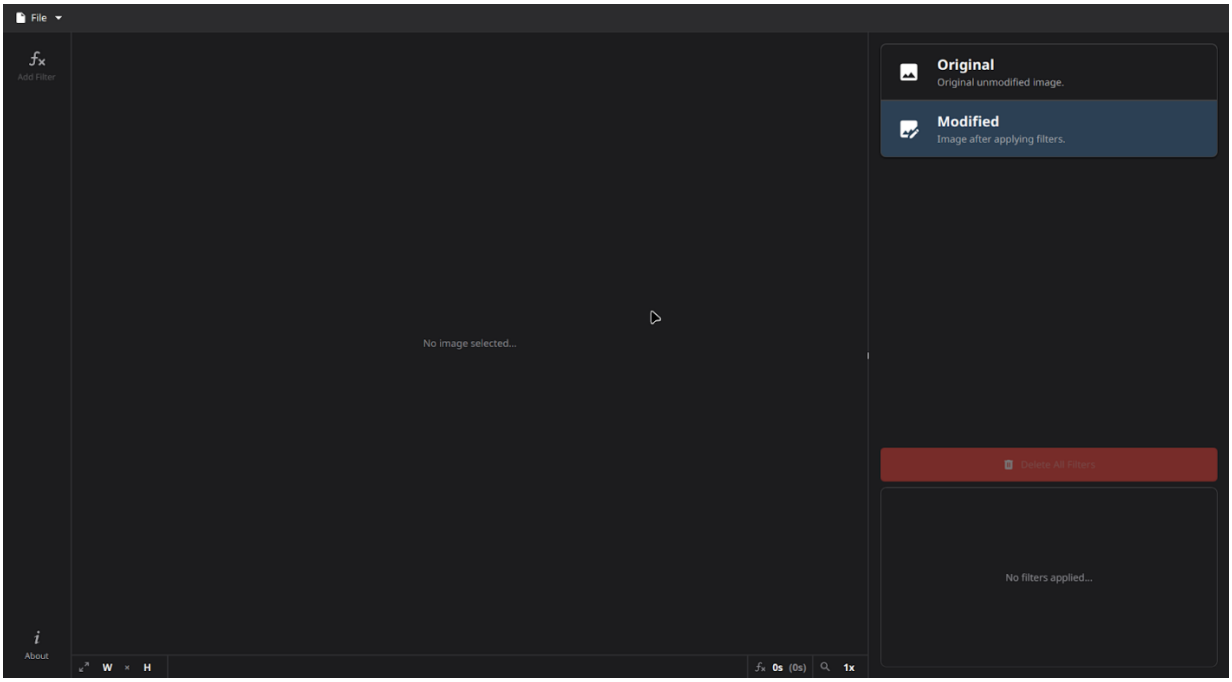
Ukázka implementace Gaussova rozostření:

```
public class GaussianBlurFilter extends KernelFilter {

    private int kernelSize;

    private int sigma;


    private double calculateGaussian(int x, int y, int sigma) {

        return Math.exp(-(x*x + y*y) / (2.0 * sigma * sigma));

    }

}
```

7. Uživatelské rozhraní (UI)

Uživatelské rozhraní je postaveno na JavaFX s využitím knihovny AtlantaFX pro moderní vzhled. Aplikace používá tmavý motiv.

7.1. Rozvržení aplikace



základní vzhled aplikace po otevření

7.2. Komponenty rozhraní

Komponenta	Popis
MenuBar	Horní lišta s menu File (Generate, Load, Save)
ActionBar	Levý panel s tlačítky pro rychlé akce
ImageViewer	Centrální plocha pro zobrazení obrázku se zoomem
Sidebar	Pravý panel pro správu filtrů
StatusBar	Spodní lišta s informacemi o obrázku
FilterLoadingScreen	Překryv zobrazující průběh aplikace filtrů

7.3. Prohlížení obrázků (ImageViewer)

Hlavní komponenta pro zobrazení a manipulaci s obrázky, umí:

- Zobrazovat aktuální obrázek (originální nebo upravený)
- Přibližovat a oddalovat pomocí Ctrl + kolečko
- Posouvat obrázek tažením myši
- Automaticky přizpůsobovat zoom při změně velikosti okna

Limity zoomu:

- Minimální: 0.1% (podle velikosti obrázku)
- Výchozí: 90%
- Maximální: 500%

7.4. StatusBar

Umí zobrazovat následující informace:

- **Rozměry obrázku** – šířka × výška (v pixelech)
    - **Zoom** – aktuální úroveň přiblížení (např. 1.00x)
    - **Čas filtrů** – skutečný čas a čas bez cache

7.5. Formulářové prvky

Aplikace obsahuje vlastní systém formulářových prvků s validací:

Prvek	Popis
TextField	Základní textové pole s validací
IntegerTextField	Textové pole pro celá čísla
DoubleTextField	Textové pole pro desetinná čísla
ChoiceField	Výběr z možností s popisem
SizeField	Dvojité pole pro rozměry (šířka × výška)
IntegerSliderField	Posuvník pro výběr hodnoty
CheckboxField	Zaškrtávací políčko

8.Popis hlavních tříd

8.1. Paintpp

Hlavní třída aplikace rozšiřující `javafx.application.Application`

8.2. ImageManager

Statická třída spravující aktuálně načtený obrázek.

- **IMAGE_PROVIDER** – aktuální zdroj obrázku
- **DISPLAYING_MODIFIED_IMAGE** – příznak zobrazení upraveného obrázku

8.3. FilterManager

Statická třída spravující seznam aktivních filtrů.

- **FILTERS** – seznam aktivních filtrů
- **FILTERED_IMAGE** – výsledný obrázek po aplikaci filtrů
- **LAST_FILTER_APPLY_TIME** – čas poslední aplikace filtrů

Při změně seznamu filtrů spustit asynchronní přepočet. Během přepočtu zobrazovat FilterLoadingScreen s průběhem.

8.4. ImageFilter (Abstraktní třída)

Základní třída pro všechny filtry. Implementovat:

- Caching – ukládat výsledek filtru pro opakované použití
- Dirty flag – označení potřeby přepočítání
- Měření času – calculationTime a copyTime

8.5. KernelFilter (Abstraktní třída)

Rozšíření ImageFilter pro konvoluční filtry. Poskytovat:

- Automatickou normalizaci jádra
- Abstraktní metodu constructKernel() pro definici konvolučního jádra
- Implementaci aplikace konvoluce

8.6. ObservableValue<T>

Generická třída pro sledování změn.

9. Závěr

Projekt Paint++ splnil svůj hlavní cíl – vytvořit funkční grafický editor s obrazovými filtry. Aplikace umožňuje načítat obrázky, aplikovat na ně různé efekty a výsledek uložit. Celkem bylo implementováno 14 filtrů, od jednoduchých (negativ, černobílá) po složitější konvoluční filtry (Gaussovo rozmazání, detekce hran). Během vývoje se ukázalo, že největší výzvou bylo zajistit plynulý chod aplikace při zpracování větších obrázků. Tento problém byl vyřešen asynchronním zpracováním filtrů na pozadí a systémem kešování, který ukládá výsledky filtrů pro opakované použití. Projekt využívá několik návrhových vzorů – Observer pro automatickou aktualizaci rozhraní, Strategy pro různé zdroje obrázků a Template Method pro jednotnou strukturu filtrů. Díky modulární architektuře je možné snadno přidávat nové filtry bez zásahu do zbytku kódu.

Aplikace je plně funkční a použitelná pro základní úpravy fotografií. Zdrojový kód je přehledně organizován do balíčků podle účelu a celý projekt je možné sestavit a spustit pomocí Mavenu.