IB111 – Domací úkol 6 (25 b)

Skupina 28 (pondělí 12:00 – 13:50)

19.12.2016 22:00

Čeká vás poslední domácí úkol, který se bude zabývat bitmapovou grafikou. Tentokrát nebudeme vytvářet hru, ale pokusíme se o vytvoření programu, který možná využijete i v praxi. V této domácí úloze dostáváte poměrně velký prostor pro vaše nápady, budete si vymýšlet i vlastní funkce. Nebojte se tedy experimentovat.

Výsledkem tohoto úkolu bude sada nástrojů pro dávkové zpracovávání obrázků. Sada nástrojů se bude skládat z několika funkcí vytvořených podle této šablony:

```
jmeno_funkce(
        jmeno_vstupniho_souboru,
        jmeno_vystupniho_souboru,
        dalsi_parametry
)
```

Jméno souboru může být celá jeho absolutní nebo relativní cesta. Každá funkce bude aplikovat právě jeden filtr na obrázek uložený ve vstupním souboru. Výsledek uloží do výstupního souboru.

Filtr zde chápeme jako funkci, která nějak modifikuje vstupní obrázek jako celek. Například funkce převádějící barevný obrázek na černobílý je filtr. Naopak štětec v grafickém programu nikoliv, protože nepracuje s celým obrázkem a vyžaduje další informace (klikání) od uživatele.

Zadání 1. část:

Vaším úkolem je vymyslet několik takových filtrů a implementovat je do jednoho `*.py` souboru, který budete odevzdávat. Za každý filtr můžete získat až 4 body, ve výjimečných případech 5 bodů. Body budu přidělovat podle složitosti zvoleného filtru. Například jednoduchý filtr, který z obrázku odstraní červený kanál, bude hodnocen maximálně třemi body. Naopak filtr, který umožní vynásobit každou barevnou složku jiným číslem (včetně nuly), je obecnější a může získat 4 body. Nespoléhejte se tedy na to, že vytvoříte pouze 5 filtrů a všechny budou hodnoceny pěti body. Triviální filtry jako například identita získají maximálně 1 bod. Doporučuji implementovat alespoň sedm filtrů.

Náměty:

Uvedu zde několik filtrů, kterými se můžete nechat inspirovat. Můžete vytvořit tyto filtry nebo si vymyslet vlastní:

- **Zmenšení nebo zvětšení obrázku:**

```
def scaleImage(inputFile, outputFile, newSizeX, newSizeY):
        # pokud je newSizeX == 0 nebo newSizeY == 0,
        # tak obrazek zachova pomer stran
        # todo ...

scaleImage("input.png", "output.png", 640, 0)
```

- **Vynásobení každého barevného kanálu konstantou:**

```
def multiplyColors(
        inputFile,
        outputFile,
        redCoef,
        greenCoef,
        blueCoef
):
        # red * redCoef, green * greenCoef, ...
        # todo ...

multiplyColors("input.png", "output.png", 1, 0.5, 0)
```

- **Detekce hran v obrázku nebo obecná konvoluce (Laplaceův filtr, Sobelův filtr, ...):**

```
def detectEdges(inputFile, outputFile):
        # barva noveho pixelu je vypocitana
        # z puvodni barvy a barvy sousedu
        # todo ...

detectEdges("input.png", "output.png")
```

- **Vytvoření bílého okraje pro obrázky vkládané do textu:**

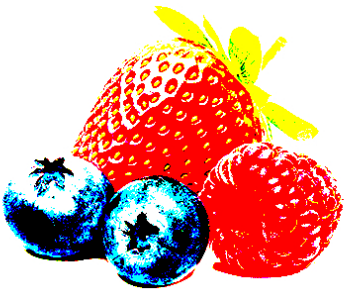
```
def addWhiteBorder(inputFile, outputFile, borderSize):
        # vytvori bily ramecek,
        # ktery plynule prechazi do obrazku
        # todo ...

addWhiteBorder("input.png", "output.png", 10)
```

- **Vytvoření jednobarevné masky pomocí prahování (vhodné například pro OCR, složitější varianta dynamicky mění hodnotu prahu na různých souřadnicích v obrázku):**

```
def applyTreshold(
        inputFile,
        outputFile,
        tresholdRed,
        tresholdGreen,
        tresholdBlue
):
        # pro kazdy kanal zvlast:
        # pokud je hodnota vetsi nez treshold,
        # zaokrouhlim na 255, jinak na nulu
        # todo ...

applyTreshold("input.png", "output.png", 128, 64, 32)
```

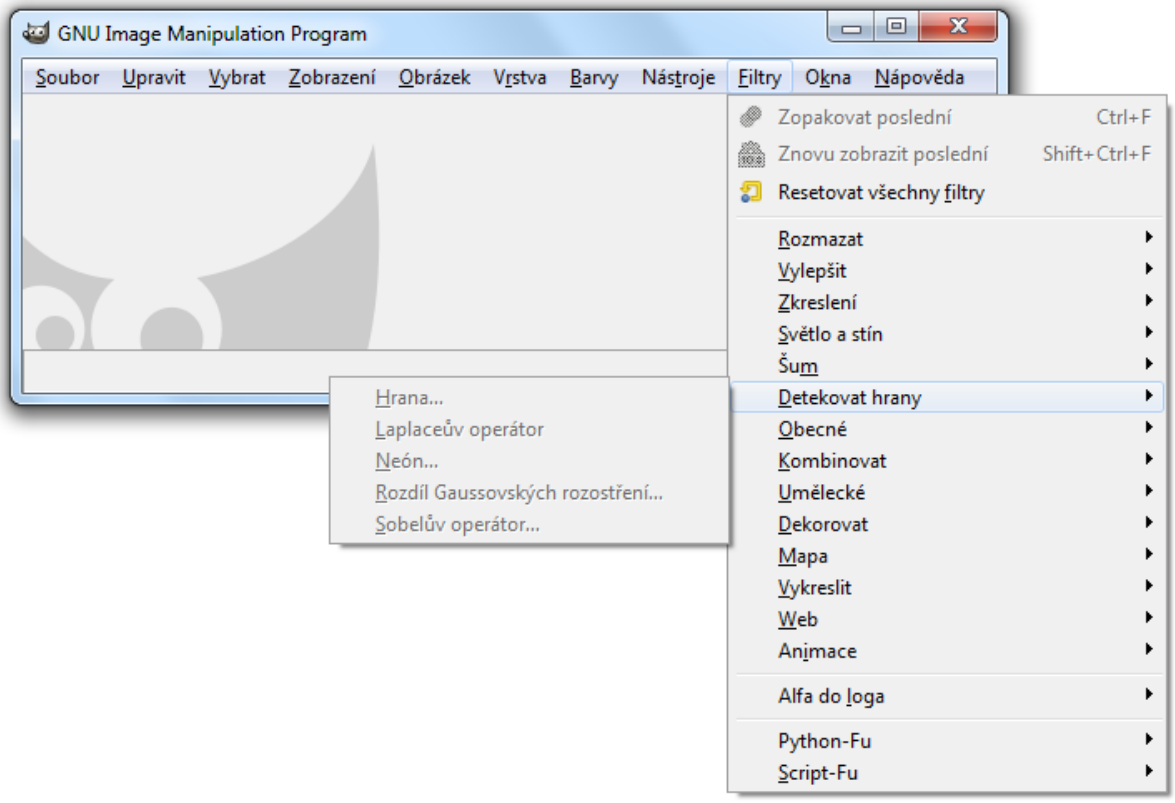


- **Cokoliv dalšího vás napadne:**

```
def contour(inputFile, outputFile):
        # todo ...

contour("input.png", "output.png")
```

Další inspiraci můžete získat třeba ze seznamu filtrů, které podporuje volně šiřitelný program GIMP:



Zadání 2. část:

Nakonec vytvořte funkci `demo(inputFileName)`, která aplikuje všechny vaše filtry na zadaný obrázek a výsledky uloží do souborů s názvy `out1.png`, `out2.png`, ... Funkce `demo` bude vypadat například takto:

```
def demo(inputFile):
        scaleImage(inputFile, "out1.png", 640, 0)
        multiplyColors(inputFile, "out2.png", 1, 0.5, 0)
        detectEdges(inputFile, "out3.png")
        addWhiteBorder(inputFile, "out4.png", 10)
        applyTreshold(inputFile, "out5.png", 128, 64, 32)
        contour(inputFile, "out6.png")
```

Tímto pro vás zadání domácí úlohy končí, ale přesto doporučuji tento text dočíst do konce.

Zajímavost:

Vaše funkce můžete spouštět z příkazové řádky stejně jako ostatní programy. Příkaz pro spuštění funkce `foo()` v souboru `soubor.py` může vypadat třeba takto:

```
python -c "import soubor; soubor.foo()"
```

Po malé úpravě můžete spouštět vaše funkce jako celé programy v jazyce Python. Například tento kód načte argumenty přímo z příkazové řádky a předá je vaší funkci:

```
import sys

print("Number of arguments:", len(sys.argv), "arguments.")
print("Argument List:", str(sys.argv))

if len(sys.argv) == 6:
    multiplyColors(
        sys.argv[1],
        sys.argv[2],
        float(sys.argv[3]),
        float(sys.argv[4]),
        float(sys.argv[5])
    )
else:
    print("Incorrent number of arguments.")
    print("Please use the following format:")
    print("muliplyColors <input> <output> \
        <red coefficient> <green coef> <blue coef>")
    print("Example:")
    print("multiplyColors input.png output.png 1 0.5 0")
```

Když bude výše uvedený kód uložený v souboru `multiplyColors.py`, tak jej můžeme zavolat přímo z příkazové řádky takto:

```
python multiplyColors.py input.jpg output.jpg 1 0.5 0
```

Zde už můžeme využít sílu příkazové řádky (terminálu) a zavolat naši funkci například pro všechny soubory v dané složce takto:

- Windows:

  ```
  for /r %i in (*) do python multiplyColors.py %i output-%i 1 0.5 0
  ```

- Unix:

  ```
  for file in /dir/*
  do
    python multiplyColors "$file" output-"$file" 1 0.5 0
  done
  ```

Bonus:

Vstupní a výstupní soubor nemusí být jen absolutní nebo relativní cesta k jednomu souboru, může se jednat o celý adresář. Pokud uživatel zadá jako vstupní soubor celý adresář, funkce zpracuje všechny obrázky v tomto adresáři a uloží je do výstupního adresáře.

Pokud je zadaný vstupní a výstupní adresář totožný nebo pokud je jako jméno výstupního souboru zadán prázdný řetězec, tak k názvu výstupního souboru přidáme prefix [`output-`].

Můžete se rozhodnout, které z bonusových úloh budete implementovat, nemusí to být nutně všechny.

Poznámka:

Závěrem jen připomenu, že řešení odevzdáváte do příslušné odevzdávárny v ISu podle vaší skupiny. Do odevzdávárny můžete svá řešení nahrát vícekrát, počítá se poslední odevzdání. Hlídejte si prosím termín i čas uzavření vaší odevzdávárny. Před termínem odevzdání za mnou můžete po domluvě přijít na konzultaci. Když mi ukážete nebo pošlete svoje řešení s předstihem, tak vás rád upozorním na případné nedostatky, za které bych ve finální verzi strhnul nějaké body. Nenechávejte to na poslední chvíli a neopisujte.