DUM č. 17 v sadě

35. Inf-11 Objektové programování v Greenfoot

Autor: Lukáš Rýdlo

Datum: 08.06.2014

Ročník: studenti semináře

Anotace DUMu: Filtr převodu obrázku na odstíny šedé, převod průměrováním a luminositou. Prahování.

evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Projekt: konzolový BMP editor v C
Filtr pro převod do odstínů šedé

Úvod

V této lekci využijeme dříve vytvořené funkce pro načtení a zapsání souboru k prvnímu grafickému efektu. Převedeme soubor do odstínů šedé. Je dobré udělat v úvodu hodiny krátkou odbočku k vysvětlení rozdílu mezi převodem na černobílou, kdy se v obrázku vyskytují pouze černé nebo bílé body a převodem do odstínů šedé (neboli grayscale), kdy obrázek obsahuje pixely, jejichž barevné složky jsou si rovny. Dostáváme tak 256 úrovní odstínů šedé, včetně bílé a černé.

Pokud by někdo ze studentů chtěl zkusit převod do černobílé, je to poměrně jednoduchá úloha, kterou by měl zvládnout každý sám, ale vyžaduje stejně nejprve spočítat jas daného pixel (tj. zjednodušeně odstín šedé). Pak se použije tzv. prahování, neboli se určí dělící hodnota (obvykle polovina rozsahu, tj. 128) a všechny pixely, jejichž šedý odstín je nižší jsou černé a vyšší bílé.

Můžeme také vytvořit jiný prahovací efekt: zvolit více úravní prahování, např. 80 a 160 a pixely, které mají jas menší než 80 obarvit třeba tmavě modře, v rozmezí 80–160 světle zeleně a nad 160 žlutě.

Při zjišťování odstínu šedě je nutné průměrovat jednotlivé barevné složky daného pixelu. Existuje několik užívaných vzorců. Obyčejné průměrování $gray = \frac{R}{3} + \frac{G}{3} + \frac{B}{3}$ nedosahuje příliš dobrých výsledků, protože lidské oko je jinak citlivé na světlost jednotlivých složek. Lepší je použít vzorce pro luminositu $gray = 0{,}21 \cdot R + 0{,}72 \cdot G + 0{,}07 \cdot B$, který bere do úvahy různé vnímání různých složek barev. Na výběr jsou další vzorce, bližší informace získají studenti např. na webu http://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/ nebo stránkách http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/ či Wikipedii.

Úkoly s řešeními

1. V souboru bmp.c vymažte celou funkci main, včetně deklarace. Vytvořte nový soubor bmp2gray.c, ve kterém vytvoříte (pouze) novou funkci main. Program se zeptá uživatele na název vstupního a výstupního souboru. Vstupní otevře a uloží pod názvem výstupního. Nezapomeňte, že používáte funkce pro čtení a ukládání BMP, které jsou deklarované v hlavičkovém souboru bmp.h. Projekt na příkazové řádce zkompilujte (uvědomte si, které všechny .c soubory nyní používáme a je nutné je proto zkompilovat společně) a spusťte. Zkompilujte zvlášť soubor bmp.c a bmp2grey.c do objektových .o souborů a teprve potom zvlášť linkujte linkerem.

   V souboru bmp2gray.c includuje bmp.h (nepoužijí se symboly většítka a menšítka, ale uvozovky, protože hlavičkový soubor není v systémových cestách, ale aktuálním adresáři). Funkce main zůstává prakticky stejná, jako v předchozí lekci. Ke kompilaci použijeme nejprve společný příkaz `gcc -o bmp2gray bmp2gray.c bmp.c`*, který kompiluje všechny zdrojové kódy znovu v jednom běhu. Poté pouze kompilujeme (bez linkování)* `gcc -c -o bmp.o bmp.c` *vznikne objektový soubor funkcí v bmp.c a pak ještě* `gcc -c -o bmp2gray.o bmp2gray.c`*, který vytvoří objektový soubor funkce main (chybí vložení funkcí z bmp.c). Oba soubory dohromady linkujeme z objektových souborů příkazem* `gcc -o bmp2gray bmp2gray.o bmp.o`*. Tento postup se hodí hlavně u větších projektů, kde kompilace trvá dlouho. Při změně jediného .c souboru stačí překompilovat tento jediný do .o souboru a pak znovu linkovat (což už je celkem rychlé). Ušetřili jsme čas kompilováním nezměněných .c souborů (např. bmp.c už měnit nebudeme). Kód mainu vypadá s vynechávkami takto:*

```
int main(int argc, char *argv[])
{
  fileHeader fh;
  picHeader  ph;
  sPixel □  image=NULL;
  int status=0;

  char infile[255];
  char outfile[255];

  printf("Zadej vstupni soubor: ");
  scanf("%s",□);
  printf("Zadej vystupni soubor: ");
  scanf("%s",□);

  status = readBMPFile(□, □, □, □);
  if(status==ERR_OK) status = writeBMPFile(□, □, □, □);

  switch(□) {
      case ERR_OK:
         printf("Ulozeno.\n");
         break;
      case ERR_INFILE:
```

```
            printf("Nelze otevrit vstupni soubor.\n");
            break;
         case ERR_NOTBMP:
            printf("Vstupni soubor neni BMP.\n");
            break;
         case ERR_ALLOC:
            printf("Nepodarilo se alokovat pamet.\n");
            break;
         case ERR_DATA:
            printf("Chyba pri cteni nebo zapisu dat. Malo mista?\n");
            break;
         case ERR_OUTFILE:
            printf("Nepodarilo se otevrit vystupni soubor.\n");
            break;
   }

   freeImage(□,□);
   getchar();

   return □;
}
```

2. Doplňte do mainu mezi čtení a zápis souboru změnu obrazových dat tak, aby byl obrázek v odstínech šedé (grayscale). Můžete použít vzorec pro průměrování nebo luminositu. Zkompilujte a zkontrolujte, zda správně funguje.

   Efekt je poměrně jednoduchý, stačí přepočítat hodnoty barevných složek a přepsat je vypočtenou hodnotou šedé. Jeho umístění je jistě jasné z toho, že obrázek musí být nejprve načtený, pak upravený a nakonec uložený. V řešení chybí deklarace proměnný a je použito průměrování:

```
for(y=0;y<□;y++) {
   for(x=0;x<□;x++) {
      g = image[x][y].blue/3+image[x][y].red/3+image[x][y].green/3;
      image[x][y].blue  = g;
      image[x][y].green = g;
      image[x][y].red   = g;
   }
}
```

3. Vytvořte nové soubory bmpfilters.c a bmpfilters.h, které budou obsahovat implementaci a deklaraci různých efektů. Nezapomeňte kód bmpfilters.h vypodmínkovat direktivou a zavedením _BMPFILTERS_H. Zatím v nich bude pouze filtr na šedou (můžete vytvořit dva různé s průměrováním a luminositou) s deklarací:

```
void makeGray(sPixel ** image, fileHeader * fh, picHeader * ph);
```

Soubor bmpfilters.h je krátký (nezapomňte includovat soubor, který je potřeba kvůli použitým datovým strukturám…):

```
#include □


#□ _BMPFILTERS_H
```

```
#define □
void makeGray(sPixel ** image, fileHeader * fh, picHeader * ph);

#endif
```

Podobně soubor bmp.c obsahuje prakticky jen vykopírovaný kód změny na šedou, ale nesmíme zapomenout, které všechny hlavičkové soubory jsou potřeba, vzhledem k použitým strukturám a funkcím… V místě ▒ je vynecháno několik řádků kódu převodu na šedou.

```
#include "□.h"
#include "□.h"

void makeGray(sPixel ** image, fileHeader * fh, picHeader * ph) {
  int x, y, g;

  for(y=0;y<ph□height;y++) {
      for(x=0;x<ph□width;x++) {
          ▒
      }
  }
}
```

4. Zkompilujte soubor bmpfilters.c do objektového kódu. Upravte bmp2gray.c tak, aby volal funkci makeGray. Linkujte celý projekt do výsledné binárky. Přemýšlejte, které všechny .c soubory je před linkováním nutné znovu překompilovat.

   Změnil se pouze soubor bmp2gray.c, takže jej musíme opět překompilovat s parametrem „-c“ do objektové podoby, než slinkujeme všechny tři soubory bmp.o, bmpfilters.o a bmp2gray.o. Volání funkce makeGray v mainu na místě původního kódu převodu do šedé je jednoduché, jen je potřeba dávat pozor na ukazatele a nezapomenout, že se smí spustit jen tehdy, pokud při načítání nenastala chyba.

Úkoly

1. V souboru bmp.c vymažte celou funkci main, včetně deklarace. Vytvořte nový soubor bmp2gray.c, ve kterém vytvoříte (pouze) novou funkci main. Program se zeptá uživatele na název vstupního a výstupního souboru. Vstupní otevře a uloží pod názvem výstupního. Nezapomeňte, že používáte funkce pro čtení a ukládání BMP, které jsou deklarované v hlavičkovém souboru bmp.h. Projekt na příkazové řádce zkompilujte (uvědomte si, které všechny .c soubory nyní používáme a je nutné je proto zkompilovat společně) a spusťte. Zkompilujte zvlášť soubor bmp.c a bmp2grey.c do objektových .o souborů a teprve potom zvlášť linkujte linkerem.

2. Doplňte do mainu mezi čtení a zápis souboru změnu obrazových dat tak, aby byl obrázek v odstínech šedé (grayscale). Můžete použít vzorec pro průměrování nebo luminositu. Zkompilujte a zkontrolujte, zda správně funguje.

3. Vytvořte nové soubory bmpfilters.c a bmpfilters.h, které budou obsahovat implementaci a deklaraci různých efektů. Nezapomeňte kód bmpfilters.h vypodmínkovat direktivou a zavedením _BMPFILTERS_H. Zatím v nich bude pouze filtr na šedou (můžete vytvořit dva různé s průměrováním a luminositou) s deklarací:

   ```
   void makeGray(sPixel ** image, fileHeader * fh, picHeader * ph);
   ```

4. Zkompilujte soubor bmpfilters.c do objektového kódu. Upravte bmp2gray.c tak, aby volal funkci makeGray. Linkujte celý projekt do výsledné binárky. Přemýšlejte, které všechny .c soubory je před linkováním nutné znovu překompilovat.