DUM č. 19 v sadě

35. Inf-11 Objektové programování v Greenfoot

Autor: Lukáš Rýdlo

Datum: 08.06.2014

Ročník: studenti semináře

Anotace DUMu: Vložení vodoznaku do souboru. Zarovnání a překrývání souborů o různých rozměrech. Průhlednost vodoznaku. Volba filtru parametrem z příkazové řádky.

esf 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 vodoznak a výběr filtru

Úvod

V této lekci vytvoříme filtr pro vkládání obrázku do obrázku, jako vodoznak a k doplnění volby filtru, který chceme aplikovat dalším parametrem na příkazové řádce. Lekce nevyžaduje žádnou novou teorii a proto můžeme ihned přistoupit k řešení úkolů.

Úkoly s řešeními

1. Vytvořte nový filtr s názvem makeWatermark, který bude vkládat druhý předaný obrázek přes první, ale maximálně po původní velikost prvního obrázku. Tj. bude-li první obrázek menší, nezmění se jeho velikost a bude zakrytý celý. Bude-li větší, bude zakryta druhým obrázkem jen jeho část. V prvním úkolu je na vás, kde a která jeho část bude umístěná. (Nejjednodušší je umístit obrázek levým dolním rohem do levého dolního rohu, tj. na pozici [0;0].) Filtr musí přebírat pochopitelně data obou (již otevřených a načtených) obrázků. Vytvořte i příslušný main v souboru bmp2watermark.c, který bude z příkazové řádky přebírat celkem 3 soubory: vstupní, s vodoznakem a výstupní.

   Řešení se může zdát komplikované, ale není. Umístíme deklaraci funkce do bmpfilters.h, deklarovaná funkce bude:

```
void makeWatermark(sPixel ** image, fileHeader * fh, picHeader * ph,
sPixel ** w_image, fileHeader * w_fh, picHeader * w_ph);
```

   Ve funkci je nutné zvážit, jaké jsou rozsahy pro x a y při vykopírovávání, jinak je řešení zřejmé:

```
void makeWatermark(sPixel ** image, fileHeader * fh, picHeader * ph,
sPixel ** w_image, fileHeader * w_fh, picHeader * w_ph) {
  int x, y;

  for(y=0;y<□->□ && y<□->□;y++) {
      for(x=0;x<□->□ && x<□->□;x++) {
          □[x][y].blue  = □[x][y].blue;
          □[x][y].green = □[x][y].green;
          □[x][y].red   = □[x][y].red;
     }
  }
}
```

   Více změn bude v mainu, který musí nyní (úspěšně a bez chyb) načítat více souborů. Důležitá část kódu s vynechávkami značenými □ je zde:

```
  fileHeader fh, wfh;
  picHeader  ph, wph;
  sPixel ** image;
  sPixel ** wimage;

  int status=0;
```

```
  if (arg□!=□) {
     printf("Chybne volani. Zadejte:\n");
     printf("%s infile.bmp watermark.bmp outfile.bmp\n\n",arg□[□]);
     return ERR_ARG;
  }

  status = readBMPFile(arg□[□], &□, &□, &□);
  if(□) status = readBMPFile(arg□[□], &□, &□, &□);
  if(□) {
      makeWatermark(image, □, □, wimage, □, □);
      status = writeBMPFile(arg□[□], &□, &□, &□);
  }

  freeImage(□,ph);
  freeImage(□,wph);
```

2. Upravte funkci vodoznaku tak, aby složený obraz vypadal opravdu jako s (barevným) vodoznakem, takže aby překrývající obrázek zastiňoval pouze částečně překrývaný. Výslednou intenzitu pixelu volte 1:2 v poměru vodoznak:původní obraz.

   Toto je jen drobná úprava funkce makeWatermark. Stačí přiřazovat do obrazu „zprůměrované“ (resp. v poměru 1:2 sečtené, což přesně nejde, použijeme zaokrouhlené hodnoty 0,4:0,6) hodnoty barevných kanálů. Vzorové řešení pro jeden barevný kanál je:

```
□[x][y].blue  = 0.6*□[x][y].blue + 0.4*□[x][y].blue;
```

3. Upravte funkci vodoznaku tak, aby vložený vodoznak byl zarovnaný na střed původního obrázku. Zkontrolujte, že funkce stále funguje pro vodoznaky menší i větší než původní obrázek.

   Úkol je spíše matematicko-logický, budeme muset upravovat hodnoty proměnných a podmínky v cyklech, kterými se prochází obrazová data. Nejjednodušší bude spočítat souřadnice středů obou obrázků a svislý a vodorovný „poloměr“, kterým je polovina výšky, resp. šířky, menšího z obrázků. Je nutné je opravdu počítat zvlášť (dvě samostatné podmínky), abychom se vyvarovali časté chyby, kdy student opomene možnost, že v jednom směru je menší první a v druhém druhý obrázek. Cyklus pro x a y poběží pro čísla z intervalu od −polorměr do +poloměr. Hodnota x a y se bude připočítávat k souřadnicím středu obrázku, který je zrovna indexovaný. Cenzurované řešení:

```
int x, y;
int icy = ph->□/2;    //image center-x
int icx = ph->□/2;    //image center-y
int wcy = w_ph->□/2;  //watermark center-x
int wcx = w_ph->□/2;  //watermark center-y
int rx  = (□<□?□:□);  //smaller radius-x
int ry  = (□<□?□:□);  //smaller radius-y

for(y=-□;y<□;y++) {
    for(x=-□;x<□;x++) {
image[□+□][□+□].blue  = 0.6*□[□+□][□+□].blue + 0.4*□[□+□][□+□].blue;
image[□+□][□+□].green = 0.6*□[□+□][□+□].green + 0.4*□[□+□][□+□].green;
```

```
image[□+□][□+□].red   = 0.6*□[□+□][□+□].red + 0.4*□[□+□][□+□].red;
    }
}
```

4. Vytvořte nový soubor bmpmain.c, který bude obsahovat úpravu stávajícího mainu. Bude přebírat na příkazové řádce jako první parametr jeden znak, který bude symbolizovat volbu efektu (g – makeGray, c – makeBlueCircle, d – makeLightCircle, w – makeWatermark). Pokud bude efekt „g“, „c“ nebo „d“, budou na příkazové řádce požadovány další dva parametry – názvy souborů. Pro efekt „w“ budou požadované názvy tři. Ve všech ostatních případech (včetně jiného písmene pro efekt) se vypíše nápověda ke spuštění a vrátí hodnota ERR_ARG. Nezapomeňte na korektní dealokaci veškeré paměti (i po souboru s vodoznakem).

   Je celá řada způsobů, jak vyřešit podmínku na správně zadané volby. Jedna z možných – kontrola jedinou rozsáhlou podmínkou ihned v začátku programu, je předvedena ve vzorové implementaci, ale pro samostatnou úvahu studentů jsou v ní některé znaky vynechány. Symbol □ nahrazuje právě jeden znak. Symbol ○ nahrazuje více znaků (proměnnou, výraz, operátor, …).

```
#include "○"
#include "○"
#include <stdio.h>
#include <stdlib.h>

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

  if ((argc□□) ||  //neither first parameter (filter)
      //filter g, c, d, needs 2 filenames
      ((arg□[□][□]=='g' || arg□[□][□]=='c' ||
        arg□[□][□]=='d') && (arg□!=□))  ||
      //filter w needs 3 filenames
      (arg□[□][□]=='w' && arg□!=□)  ||
      //not known filter
      (arg□[□][□]!='g' && arg□[□][□]!='c' &&
       arg□[□][□]!='d' && arg□[□][□]!='w')
     ) {
     printf("Chybne volani. Zvolte:\n");
     printf("%s g in.bmp out.bmp - prevod do odstinu sede\n",arg□[□]);
     printf("%s c in.bmp out.bmp - kresleni modreho kruhu\n",arg□[□]);
     printf("%s d in.bmp out.bmp - kresleni sveleho kruhu\n",arg□[□]);
     printf("%s w in.bmp out.bmp watermark.bmp - vlozeni vodoznaku na
stred\n\n",arg□[□]);
     return ERR_ARG;
  }
```

```
  status = readBMPFile(arg□[□], ○, ○, ○);
  if(○) {
      switch (arg□[□][□]) {
         case 'g':
           makeGray(image, &fh, &ph);
           break;
         case 'c':
           makeBlueCircle(image, &fh, &ph);
           break;
         case 'd':
           makeLightCircle(image, &fh, &ph);
           break;
         case 'w':
           status = readBMPFile(arg□[□], &wfh, &wph, &wimage);
           if (○) makeWatermark(image, &fh, &ph, wimage, &wfh, &wph);
           freeImage(○,wph);
           break;
      }
  }

  if (○) status = writeBMPFile(arg□[□], &fh, &ph, &image);

  switch(status) {
      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(○,ph);
  return status;
}
```

Úkoly

1. Vytvořte nový filtr s názvem makeWatermark, který bude vkládat druhý předaný obrázek přes první, ale maximálně po původní velikost prvního obrázku. Tj. bude-li první obrázek menší, nezmění se jeho velikost a bude zakrytý celý. Bude-li větší, bude zakryta druhým obrázkem jen jeho část. V prvním úkolu je na vás, kde a která jeho část bude umístěná. (Nejjednodušší je umístit obrázek levým dolním rohem do levého dolního rohu, tj. na pozici [0;0].) Filtr musí přebírat pochopitelně data obou (již otevřených a načtených) obrázků. Vytvořte i příslušný main v souboru bmp2watermark.c, který bude z příkazové řádky přebírat celkem 3 soubory: vstupní, s vodoznakem a výstupní.

2. Upravte funkci vodoznaku tak, aby složený obraz vypadal opravdu jako s (barevným) vodoznakem, takže aby překrývající obrázek zastiňoval pouze částečně překrývaný. Výslednou intenzitu pixelu volte 1:2 v poměru vodoznak:původní obraz.

3. Upravte funkci vodoznaku tak, aby vložený vodoznak byl zarovnaný na střed původního obrázku. Zkontrolujte, že funkce stále funguje pro vodoznaky menší i větší než původní obrázek.

4. Vytvořte nový soubor bmpmain.c, který bude obsahovat úpravu stávajícího mainu. Bude přebírat na příkazové řádce jako první parametr jeden znak, který bude symbolizovat volbu efektu (g – makeGray, c – makeBlueCircle, d – makeLightCircle, w – makeWatermark). Pokud bude efekt „g“, „c“ nebo „d“, budou na příkazové řádce požadovány další dva parametry – názvy souborů. Pro efekt „w“ budou požadované názvy tři. Ve všech ostatních případech (včetně jiného písmene pro efekt) se vypíše nápověda ke spuštění a vrátí hodnota ERR_ARG. Nezapomeňte na korektní dealokaci veškeré paměti (i po souboru s vodoznakem).