Képfeldolgozó nagyházi feladat

Szükséges segédkönyvtárak

Grafikus megjelenítőkönyvtárként az InfoC-n elérhető SDL könyvtárat használtam, így a programot InfoC SDL2 projektként kell létrehozni.

A program felépítése

A program 6 forrásfájlból és 5 header fájlból áll. A loadimage.c fájlban található a képek betöltését végrehajtó függvény, a writeimage.c fájlban található a képek mentését vezérlő függvény. A shading.c fájlban találhatók az árnyalatmódosító függvények, az imgmanipulate.c fájlban pedig a képet lényegesen megváltoztató függvények. A main.c-ben található a menü felépítése, a kattintásérzékelés, valamint az eszerint történő függvényhívások.

PPM fájlok kezelése

A PPM fájlok (portable pixel map) tömörítetlen adatot tárolnak egy adott képről. A fájl elején egy magic number található, amely PPM esetén mindig P6. A magic number után egy whitespace karakter következik, majd esetleges kommentek. A kommentek után következik a kép szélessége, egy decimális értékben, majd újabb whitespace. Ezután a kép magassága jön, szintén whitespace-el követve. A kép utolsó jellemzője pedig a színek maximum értéke, amely RGB színkód esetén mindig 255.

Az alapvető információk után egy whitespace-t követve a kép pixeladata következik. Mivel a maximum értéke 255 lehet, ezért minden egyes pixel r,g,b adata (amelyek egymást követve lineárisan helyezkednek el) egy bájtnyi helyet foglal a bináris fájlban. Így a beolvasás a megfelelő r,g,b változókba szintén bájtonként történik.

A fájlformátum kezelése a következő: megnyitjuk a fájlt, majd beolvassuk a magic numbert. Ellenőrizzük, hogy utána már a szélesség adata következik, vagy a következő sor elején kommentet találunk-e. Ha kommentet találunk azt egy “#” jelzi, amely komment a következő sorig tart. Ha eljutottunk a sor végére, akkor a következő sor első karakterét ellenőrizzük, ha komment akkor addig folytatjuk ezt az eljárást ameddig nem komment jön a következő sorban. Ha nem komment a következő sor első karaktere, akkor azt már a szélesség változóba olvashatjuk be. Ezután a következő adatot, ami a kép magasságára vonatkozik a magasság változóba beolvassuk. Ezek után a maximum értékkel ugyanígy teszünk, a maxval változóba. Ha ezen átjutott a program, akkor következik a kapott értékek alapján történő dinamikus memóriafoglalása egy két dimenziós tömbnek, amely “pixel” elnevezésű struktúrákat tartalmaz, amely reprezentálja a kép sor/oszlop felbontását. A kép magasságával megegyező sort, és a kép szélességével megegyező oszlopot foglalunk a memóriában. Ha a foglalás sikeres, a program a megfelelő pixelek értékeit beírja a kétdimenziós tömb megfelelő tagjaiba. Egy pixel struktúra tartalmazza az adott pixel r,g,b értékeit, integer változókként. A kapott értékeket beírva a “PPMimage” struktúrába (amely tartalmaz egy szélesség, magasság, maxval integer változót, valamint a kép pixeleinek adataiból létrehozott kétdimenziós tömböt) megkapjuk a beolvasott képet reprezentáló “image” PPMimage típusú adatot. Az imagereader függvény ezt adja a vissza a főprogramba.

PPM fájl mentése úgy történik, hogy a megadott formátum szerint visszaírjuk egy új fájlba az új kép adatait. Először a magic number, majd a szélesség, magasság, maxval és ezek után a kapott pixeldata.

PPM fájlok manipulálása

A program képes egy adott PPM képfájl fényerejét növelni, csökkenteni, kontrasztot növelni, csökkenteni, negatívot létrehozni, egy rózsaszín filterrel változtatni a képet, vízszintesen és függőlegesen tükrözni, valamint a mostanában az interneten elterjedt “deep fry” memenek megfelelő filtert is tud alkalmazni a képen.

A kép megváltoztatásához a kép összes pixelén el kell végezni valamilyen műveletet. Erre a PPMimage struktúra PPMpixeldata adattagját használjuk.

A fényerő 10%-os növeléséhez minden r,g,b értéket 1.1-el meg kell szorozni, ügyelve arra, hogy az rgb értékek 0 és 255 között maradjanak. Hasonlóan 10%-os fényerő csökkentéshez minden rgb értéket 0.9-el kell szorozni.

A kontraszt növeléséhez és csökkentéséhez az alábbi algoritmust érdemes használni:

```
factor = (259 * (contrast + 255)) / (255 * (259 - contrast))
colour = GetPixelColour(x, y)
newRed = Truncate(factor * (Red(colour) - 128) + 128)
newGreen = Truncate(factor * (Green(colour) - 128) + 128)
newBlue = Truncate(factor * (Blue(colour) - 128) + 128)
```

forrás: http://thecryptmag.com/Online/56/imgproc_5.html

Ahol a contrast változó ha pozitív, akkor növeljük a kontrasztot, ha negatív akkor csökkentjük (-255 és +255 között mozoghat), valamint a Truncate függvény figyel arra, hogy az rgb értékek 0 és 255 között maradjanak.

A kontrasztot, valamint a fényerőt módosító függvények a **shading.c** forrásfájlban találhatók.

A képről negatív létrehozása az egyik legegyszerűbb művelet: minden r,g,b értékhez a skála vele ellentétes értékét kell venni; azaz ki kell vonni 255-ből az adott r,g,b értéket, így például a 0,0,0-ból 255,255,255 lesz.

Ha képet eltoljuk a skála vörös oldala felé, akkor egy rózsaszín filtert kapunk, amelyet a programban úgy valósítottam meg, hogy a vörös értékhez hozzáadtam 85-öt (ügyelve arra, hogy a skálán belül maradjanak az értékek), valamint a g és b értékekből kivontam 20-at (szintén ügyelve a 0 és 255 határokra).

Ha a képet tükrözni szeretnénk, akkor az adott pixel adatot fel kell cserélni a vele szimmetrikusan elhelyezkedő pixel értékével. Ezt meg lehet tenni sor és oszlop szerint, függőleges és vízszintes tükrözést kapva.

A deep fry filterhez ismételve kell alkalmazni a kontraszt,pink,negatív, valamint a fényerő növelését. Hogy a végén eltüntessem a zöld filtert, amelyet fehér hátterű képeknél kaptam még egymás után 7-szer növelem a kontrasztot.

A negatív, pink, tükröző, valamint a deep fry függvény az **imgmanipulate.c** forrásfájlban találhatók.

Függvények

- Az imgmanipulate.c source függvényei
  - void negative (PPMimage image)
    Egy kapott PPMimage képben a PPMpixeldata adatokat felcseréli a 0,255 skálán vett „ellentett” értékükkel.
  - void pink (PPMimage image)
    Egy kapott PPMimage képben a PPMpixeldata adatokat eltolja a skála vörös oldala felé, így egy rózsaszín filtert kapva.
  - void mirrorhorizontal (PPMimage image) és void mirrorvertical (PPMimage image)
    Egy kapott PPMimage képben adatait tükrözi a vízszintes/függőleges tengelyre, úgy, hogy a kétdimenziós tömb megfelelő adatait felcseréli.
  - void deepfry (PPMimage image)
    A programban szereplő összes többi függvényt használja egyszerre egy megadott sorrendben, a kívánt hatáshoz sok kontrasztnövelésre, fényerő módosításra, valamint pár negatív váltásra van szükség.
- A loadimage.c source függvényei
  - PPMimage imagereader (char* filename)
    A függvény egy stringet kap paraméterként, amely string alapján megkísérli egy ppm fájl megnyitását, a PPM fájlformátum specifikációja szerint. Ha nem sikerül megnyitni a fájlt (nem szerepel a fájl a mappában vagy nem helyes a fájlnév), akkor egy üres képet ad vissza, ha pedig sikerül megnyitni a fájlt, akkor feltölti a PPMimage image változó adatstruktúráját, és ezt adja vissza.
- A shading.c source függvényei
  - void darker (PPMimage image) és void lighter (PPMimage image)
    A két függvény a kapott PPMimage képet sötétíti/világosítja, azáltal, hogy a PPMpixeldata adatokat megszorozza 0.9-el 10%-os sötétítéshez, valamint 1.1-el 10%-os világosításhoz. A lighter függvénynél szereplő truncate segédfüggvény biztosítja azt, hogy az adatok a 0,255 tartományban maradjanak.
  - void morecontrast (PPMimage image) és void lesscontrast (PPMimage image)
    A két függvény a kapott PPMimage kép kontrasztját növeli/csökkenti a fentebb megadott módon, a PPMpixeldata módosításával.
- A sub.c source függvényei és a sub.h header adatszerkezet definíciói
  - int truncate (float x)
    Egy kapott float változóról eldönti, hogy a 0,255 intervallumon van-e, és ha nincs benne, akkor az intervallum szélére teszi, attól függően, hogy az érték az intervallumhoz képest hol helyezkedik el, majd a floatot integerként adja vissza.
  - pixel struktúra
    Tartalmazza egy pixel 3 adatát; a red, green és blue értékeit.
  - PPMimage struktúra
    Tartalmazza a ppm definíciója szerint megadott fájlformátumot meghatározó értékeket. A szélességet, magasságot, a színek maximum értékét, valamint a pixel adatokat, amelyet egy kétdimenziós, pixel típusú tömbben tárol el.
- A writeimage.c source függvényei
  - void imagewriter (char* filename, PPMimage image)
    A függvény egy stringet és egy PPMimage képet kap paraméterként. A függvény létrehozza a string által meghatározott nevű fájlt, és beleírja a PPMimage adatait, a ppm definíciójának megfelelően.