Proiect Aplicații Web cu suport Java

Decrease Color Depth Gray-Scale Image

Proiect realizat de: Bordeianu David

Facultatea de Automatica si Calculatoare

Grupa: 332AB

Cuprins

1. Introducere

Ca obiectiv, proiectul a presupus crearea unei aplicații folosind Java, pentru a procesarea de imagini. Tema aleasa de mine a fost descreșterea profunzimii culorii, pentru o imagine Gray-scale. Aplicația rulează folosind cu exactitate locația (calea) imaginii dorite a fi supusa procesului de conversie. De asemenea, rezultatul conversie trebuie specificat si el, de la tastatura.

2. Descrierea implementării

Imagine: Reprezintă o clasa abstracta, ce implementează metoda din Interfata_Proiect. Nivelul acesteia de moștenire este 1.

Interfata_Proiect: Interfața propriu-zisa, implementata de clasa Imagine, ce conține apelarea către metoda de citire a imaginii.

Dimensiuni_Imagine: Este clasa ce implementează metodele pentru datele imaginii, anume, afișare setare si returnarea înălțimii si lățimii imaginii. Aceasta are nivelul de moștenire 2.

Citire_IMG: Aceasta este o clasa in care se fac constructorul si metodele de citire propriu-zise, pentru imagine, respectiv pentru citirea fișierului din care imaginea este preluata. Nivelul acesteia de moștenire este 3.

Proiect: Reprezintă „main”-ul proiectului. Este clasa in care sunt preluate calea fișierului si numele imaginii ce urmează a fi procesata, calculând simultan timpii necesari fiecărui proces, prin care trece imaginea, in conversia sa.

Consumer: Reprezintă un obiect de tip Thread, necesar implementării multithreading-ului. Este folosit pentru conversia threadului respectiv.

Producer: Cel de al doilea obiect de tip Thread care se ocupa cu apelarea metodei de citire. Aici am utilizat notify(), pentru a asigura așteptarea lui Consumer, pana când Producer citește imaginea data.

3. Algoritmii de conversie propriu–zisi

Pentru a converti imaginea color in alb negru, am folosit metoda average. Astfel, procesul rezultat reda tonul de gri, realizat prin compunerea valorii rezultate a fiecărei dintre cele trei culori, la media aritmetica a acestora.

In procesul de conversie al imaginii, am utilizat funcția următoare:

```
for(int i = 0; i < inaltime; i = i + 1)
{
    for(int j = 0; j < latime; j = j + 1)
    {
        int x = imagine.imagine.getRGB(j,  i);

        int y = (x >> 24) & 0xFF;
        int r = (x >> 16) & 0xFF;
        int g = (x >> 8) & 0xFF;
        int b = (x) & 0xFF;

        r = r & 0xC0;
        g = g & 0xC0;
        b = b & 0xC0;

        // Setam noul RGB
        x = (y << 24) | (r << 16) | (g << 8) | b;

        imagine.imagine.setRGB(j, i, x);
    }
}
```

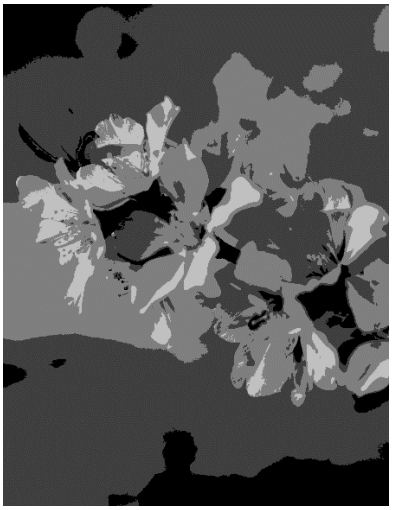
Iar funcția, pentru Grey Scale este:

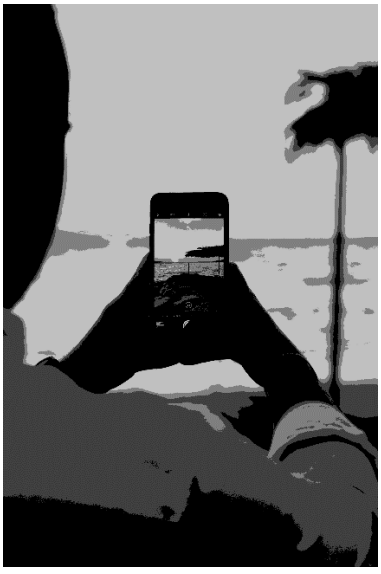
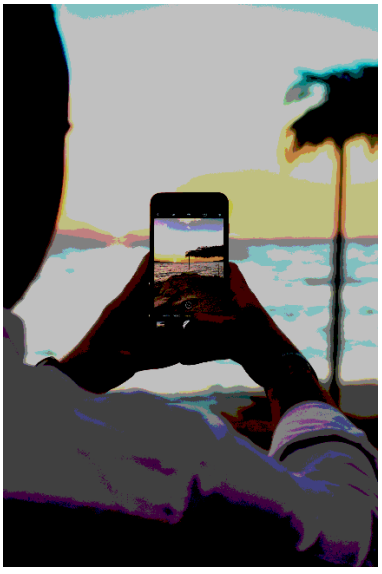
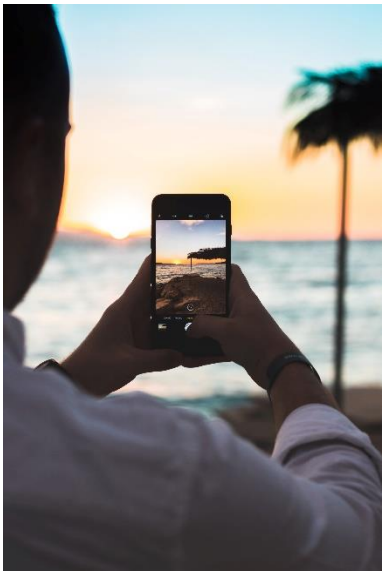
```
for(int i = 0; i < inaltime; i = i + 1)
{
    for(int j = 0; j < latime; j = j + 1)
    {
        int x = imagine.imagine.getRGB(j,  i);

        int y = (x >> 24)&0xff;
        int r = (x >> 16)&0xff;
        int g = (x >> 8)&0xff;
        int b = (x) & 0xff;

        // Calculam media
        int med = (r + g + b)/3;

        // Si modificam valorile
        x = (y << 24) | (med << 16) | (med << 8) | med;

        imagine.imagine.setRGB(j, i, x);
    }
}
```

4. Exemple

Câteva exemple, atât pentru funcția completa Grey Scale + Depth, dar si pentru a ilustra funcționalitatea funcției de luminozitate sunt:

(Puse in ordinea GS + D, D, Original)



5. Rularea

In cadrul procesului de rulare, acestea sunt evenimentele care iau loc:

```
Introduceti numele sau adresa imaginii ce va fi salvate. (*.bmp)
C:\\Users\\borde\\OneDrive\\Desktop\\Poli\\A3\\S1\\AWJ\\Proiect\\Proiect\\Output\\Ex2.bmp
Introduceti numele sau adresa imaginii pentru conversie. (*.bmp)
C:\\Users\\borde\\OneDrive\\Desktop\\Poli\\A3\\S1\\AWJ\\Proiect\\Proiect\\Input\\poza.bmp
Timpul total de executie (Citire Fisier Intrare): 3.7802398
Timpul total de executie (la constructorul fara parametrii): 0.1914212
Timpul total de executie (Procesare Poza): 7.921394299
Conversia s-a finalizat.
Timpul total de executie (Main): 15.473954701
```

6. Cerințe

In cadrul proiectului, am acoperit următoarele cerințe:

1. Imaginea sursa este întotdeauna BMP.
2. Sunt prezente in totalitate conceptele POO – încapsulare (Structura pe clase), moștenire, polimorfism, abstractizare.
3. Codul sursa respecta „coding standards” si este comentat.
4. Sunt prezente doar algoritmi/secvențe de cod low-level.
5. Operații de lucru cu fișiere.
6. Operații de intrare de la tastatura si prin parametri liniei de comanda pentru asignarea fișierelor de intrare, parametri / setările / opțiunile de execuție si pentru asignarea fișierelor de ieșire.
7. Aplicația este multimodulară.
8. Aplicația include varargs (Pentru calculul timpului unei etape din cod).
9. Aplicația include constructori.
10. Aplicația include bloc static de inițializare.
11. Aplicația include interfață, si clasa care o implementează.
12. Sunt prezente clase abstracte si clasele aferente acestora.
13. Excepțiile sunt tratate.
14. Aplicația conține 2 pachete: Pachetul1 (Pentru clasa main) si Pachetul2 (Ce conține restul claselor).
15. Aplicația conține Producer-Consumer.
16. Aplicația conține comunicație prin Pipes.

7. Bibliografie

https://docs.oracle.com/cd/E17802_01/products/products/java-media/jai/forDevelopers/jai-apidocs/com/sun/media/jai/codec/PNGDecodeParam.html
https://blog.idrsolutions.com/how-to-read-and-write-images-in-java/
https://nisal-pubudu.medium.com/understanding-threads-multi-threading-in-java-6e8c988d26af
https://stackoverflow.com/questions/35317789/java-multithreading-on-very-small-image-taking-a-long-time
https://www.itcsolutions.eu/ro/2011/07/31/tutorial-java-scjp-18-blocuri-de-initializare/
https://www.baeldung.com/java-varargs