Implementatieplan Vision Week 1

Job Verhaar & Thijs Hendrickx 11-04-2018

Doel

Voor dit project zijn er twee implementatiedoelen. Ten eerste zullen er Image Shells geschreven worden voor RGB en Grayscale afbeeldingen. Het doel van deze image shells is om elke pixel van een afbeelding op te slaan, zodat daarmee gewerkt kan worden door andere methoden in de applicatie. Ten slotte zal er ook een methode worden geschreven om een RGB image shell te converteren naar een Grayscale image shell. Dit is van belang omdat er makkelijker operaties uitgevoerd kunnen worden op een grayscale afbeelding dan op een RGB afbeelding.

Methoden

Voor de image shells is het van belang om elke pixel waarde van de afbeelding op een geoptimaliseerde manier op te slaan. Dit zal worden gedaan doormiddel van een array, vector of list. Er zal onderzoek gedaan moeten worden naar de meest geoptimaliseerde keuze voor deze applicatie. Voor de grayscale conversie kan er gekozen worden tussen implementatie d.m.v. constructor, lokale methode in de RGB of Grayscale image shells of een externe methode. Ten slotte bestaan er drie hoofdmethoden voor kleur naar grayscale conversie, namelijk Lightness, Luminosity en Average. Hier zal een keuze uit moeten worden gemaakt.

Keuze

Voor de methode van pixel opslag is gekozen een standaard GCC **Array** te gebruiken. Een list heeft meer controle over de ruimte gereserveerd voor opgeslagen objecten, maar heeft lagere prestatie snelheden dan vector en array. Het verschil tussen een vector en array is niet groot, maar er zijn een aantal nadelen aan een vector waardoor de keuze is gemaakt om een array te gebruiken. Ten eerste initialiseerd een vector alle ruimte die hij gealloceerd krijgt op 0. Een array doet dit niet, wat tijd bespaart. Verder is een standaard GCC array sneller met het opslaan van simpele objecten, zoals een struct bestaande uit een aantal karakters. Vectors zijn sneller met reken operaties, maar dat is voor deze use-case niet van belang.

Voor de grayscale conversie is gekozen om een **externe methode** te implementeren omdat de structuur van het project het lastig maakt nieuwe methoden toe te voegen aan de Student klassen. Het zou betekenen dat er een dummy implementatie geschreven zou moeten worden voor de private klassen, omdat deze van dezelfde interface afhangen. Hierdoor is gekozen een methode in de main.cpp te schrijven voor de conversie.

Ten slotte is er gekozen voor het **Luminosity** grayscale conversie algoritme. Dit algoritme is ontworpen om rekening te houden met de perceptie mogelijkheden van het menselijk oog. Onze ogen zijn gevoeliger voor groen licht, en dit is meegenomen in de berekening. Dit leidt tot accurate representatie van de afbeelding in grayscale, met een contrast dat zoveel mogelijk hetzelfde blijft als de originele afbeelding. Het Lightness algoritme brengt het contrast omlaag, en Average zit hier wat tussenin. Aangezien er geen overeenkomst is over de beste waardes voor de Luminosity grayscale conversie is er voor dit project kozen voor de orginele ITU—R gedefinieerde waardes:

(Red * 0.2126 + Green * 0.7152 + Blue * 0.0722)

Implementatie

De image shells zijn op dezelfde manier geïmplementeerd, met de twee verschillende type pixels (een struct met R G B karakters voor de RGB image shell en een karakter voor de intensity voor de grayscale image shell). Hier zijn tweedimensionale arrays aan toegevoegd om een pixel op te slaan in hoogte/wijdte formaat. Hierna zijn er methodes en constructors geïmplementeerd voor het creëren van een image shell, het zetten van pixels en het ophalen van pixels.

Voor de grayscale conversie is er een methode in main.cpp geïmplementeerd genaamd convert. Deze geeft een IntensityImage terug, en krijgt een reference naar een RGBImage mee. De methode maakt een nieuwe IntensityImage aan met dezelfde hoogte en wijdte als de RGBImage, en zet alle pixels van de RGBImage om naar grayscale met het Luminosity algoritme. Hierna geeft de methode de IntensityImage terug.

Evaluatie

Om de functionaliteiten van de geïmplementeerde Image Shells te testen zullen er twee experimenten gedaan worden. Het eerste experiment gaat dieper in over de keuze van het type opslag voor de pixels. Hier worden een standaard GCC Array en Vector implementatie vergeleken qua snelheid om zo te bepalen wat het beste is om te gebruiken. Verder zal hier ook de snelheid worden vergeleken met de snelheid van de basis implementatie, om te bepalen of de gekozen implementatie methode dicht bij de basis implementatie zit. Ten slotte wordt er ook een experiment gedaan om de correctheid van de gekozen RGBImage implementatie te testen en vergelijken met de basis implementatie. Hier zal gekeken worden of de informatie van een afbeelding correct wordt opgeslagen, specifiek de afmetingen en de pixels. Door deze experimenten zal kunnen worden bepaalt of de door de studenten geïmplementeerde image shells en conversie algoritme voldoen om verdere operaties uit te voeren.