# 1. Implementatieplan imageshell en grayscale

## 1.1. Namen en datum

Deniz Lektemür - 1722313

Datum: 20-2-2019

## 1.2. Doel

Het doel is om een imageshell te maken voor zowel een RGB afbeelding als voor een Intensity (gray) afbeelding die zo snel mogelijk werkt. Ook is de bedoeling om een conversie te maken van een RGB afbeelding naar een Intensity afbeelding die ervoor zorgt dat de volgende image processing stappen zo goed mogelijk verlopen.

## 1.3. Methoden

**Imageshell:**

Voor de imageshell is de grote keuze waar de pixels in opgeslagen gaan worden. De keuzes die ik overwogen heb zijn de std::vector, std::array en de C stijl array. Het voordeel van de std::vector is dat de vector de mogelijkheid heeft om uit te breiden maar aangezien de grootte van het plaatje al bekend is, is dit niet echt nodig. Het nadeel van een vector is dat het in verhouding wat slomer is. Het voordeel van een std::array is dat het in verhouding redelijk snel is. Het nadeel is echter dat een array niet uitbreidbaar is, maar aangezien we de grootte van het plaatje al weten is dit niet zo een groot probleem. Het voordeel van een C stijl array is dat het net iets sneller is dan een std::array. Het nadeel is dat het lastiger is om het geheugen te beheren.

Naast de keuze voor welke container er gebruikt gaat worden is het ook belangrijk om te kiezen of er een 1-D of 2-D container gebruikt gaat worden. Het voordeel van een 1-D container is dat het snellere acces heeft en minder geheugen in beslag neemt. Het nadeel van de 1-D container is dat het geheugen in 1 groot blok staat en dat het wat lastiger is om het geheugen in te beelden. Het voordeel van de 2-D container is dat het wat makkelijker is om erover na te denken en dat het wat beter te verspreiden is over het geheugen. Maar het nadeel van een 2-D container is dat het net wat slomer is en iets meer geheugen in beslag neemt.

**Grayscale:**

Er zijn een aantal verschillende manieren om een grayscale te implementeren.

- Averaging
  **Gray = (r + g + b) / 3**
  Bij averaging wordt het gemiddelde genomen van de r, g en b waardes en die ingevuld voor de grijswaarde. Werkt redelijk goed en snel maar houdt geen rekening met het menselijk oog.

- Luma (luminance)
  **Gray = r * 0.3 + g * 0.59 + b * 0.11**
  Bij luma worden de r, g en b waardes vermenigvuldigd met constanten en vervolgens worden deze waardes bij elkaar opgeteld en ingevuld voor de grijswaarde. Houdt rekening met het menselijk oog maar werkt net wat minder snel.

- Decomposition (maximum minimum)
  **Gray = max(r, g, b)**
  **Gray = min(r, g, b)**
  Bij decomposition wordt van de r, g en b waardes de hoogste of laagste waarde gekozen (ligt eraan of er maximum of minimum decomposition gebruikt wordt) en ingevuld voor de grijswaarde. Deze methode wordt meer gebruikt als artistiek effect.

- Single color channel
  **Gray = r**
  **Gray = g**
  **Gray = b**
  Bij single color channel wordt r, g of b waarde gekozen en ingevuld voor de grijswaarde. Werkt heel snel aangezien er niks berekend hoeft te worden maar kan hele rare effecten hebben en ziet er niet altijd even goed uit.

## 1.4. Keuze

Ik heb voor de imageshell uiteindelijk gekozen om een 1-D C stijl array te gebruiken omdat dit de snelste manier is om pixels op te slaan en te gebruiken. Voor de grayscale heb ik gekozen om luma te gebruiken omdat dit algoritme rekening houdt met het menselijk oog. En aangezien luma het “mooiste” plaatje genereerd zal het waarschijnlijk makkelijker gaan met de volgende processing stappen op de afbeelding.

## 1.5. Implementatie

In de RGBImageStudent klasse ga ik een pointer in de private variabelen aanmaken. De pointer zal wijzen naar een 1-D lijst van opeenvolgende pointers naar de RGB

pixels. Hetzelfde zal ik gaan doen in de IntensityImageStudent klasse maar dan met Intensity pixels in plaats van RGB pixels.

Voor de functie getPixel(x, y) en setPixel(x, y) zal ik gebruik maken van het algoritme: y * width + x om de positie in de 1-D array te bepalen.

In de StudentPreProcessing klasse zal ik het grayscale algoritme gaan schrijven.

## 1.6. Evaluatie

Voor de imageshell zal ik gaan timen om te kijken of mijn implementatie van imageshell sneller is dan de default implementatie. En voor de grayscale zal ik de afbeeldingen van de default implementatie en mijn implementatie naast elkaar zetten om te kijken welke processing stappen beter zijn gegaan.