# Implementatieplan ImageShell

## Namen en datum

Tim van Dijk

Victor Adamse

12/4/2019

## Doel

Het maken van een ImageShell voor RGB en voor Intensity images.

## Methoden

**C-Style Array**

Voordelen:

* Built-in
* Statische lengte / Dynamische lengte d.m.v. een pointer door opnieuw te alloceren
* Staat dus op de stack of heap

Nadelen:

* Memory dien je zelf te allocaten en te deallocaten
* Gevaarlijker in gebruik want je moet alles er omheen zelf schrijven

**Std::vector**

Voordelen:

* Dynamische lengte
* Memory wordt geregeld, hoef je dus niet zelf te doen
* Veel extra functies die het gebruik makkelijk maken, denk aan push\_back, pop\_back, etc.
* Veiliger in gebruik dan een C-Style Array
* Gebruik van de STL gaat makkelijk

Nadelen:

* In essentie een dynamische C-Style array, er is dus meer overhead en een kleine hoeveelheid snelheidsafname
* Staat op de heap

**Std::array**

Voordelen:

* Staat op de stack
* Veiliger in gebruik dan een C-Style Array
* Sneller dan een std::vector
* Gebruik van de STL gaat makkelijk

Nadelen:

* Statische lengte
* Het is een wrapper voor een C-Style Array, en zal dus iets trager zijn dan het directe gebruik van een C-Style Array

**Bronnen:**  
<https://stackoverflow.com/questions/15079057/arrays-vs-vectors-introductory-similarities-and-differences>

<https://stackoverflow.com/questions/4424579/stdvector-versus-stdarray-in-c/4424658#4424658>

## Keuze

Wij hebben voor onze implementatie een Dynamische C-Style Array gekozen. Aangezien we de image moeten kunnen veranderen is er een dynamische container nodig. Ook is snelheid van belang omdat wij de default implementatie moeten evenaren of overtreffen. Verder zullen wij de STL niet nodig gaan hebben in onze implementatie. De Dynamische C-Style Array is in dit geval de snelste optie, echter is er wel meer alertheid van de programmeur nodig om dit te implementeren, aangezien memory handmatig moet worden geallocate en gedeallocate.

## Implementatie

De RGB ImageShell gaan wij implementeren in de RGBImageStudent.h en RGBImageStudent.cpp.

De Intensity ImageShell gaan wij implementeren in de IntensityImageStudent.h en IntensityImageStudent.cpp.

De implementaties hiervan zijn verder niet veel werk en code, bij de constructors dient memory te worden geallocate voor de array. Bij de set functies dient de voorgaande array te worden gedeallocate en een nieuwe array met de nieuwe width en height te worden geallocate.

De set en get functies spreken voor zich en vereisen weinig code.

## Evaluatie

Om onze methode te vergelijken met de default methode gaan we tests uitvoeren. We gaan de 2 implementaties vergelijken op snelheid. Hiervoor gaan we de RGB imageshell inlaadfunctie testen. We gaan dus kijken naar de snelheid van het inladen van een plaatje in de verschillende classes. We zullen dit proces in een loop herhalend uitvoeren. Uit de metingen zullen we de minimale, maximale en gemiddelde tijd halen. De resultaten van de default implementatie en onze implementatie zullen we vergelijken om het verschil te zien.