1. Implementatieplan opslag

1.1. Namen en datum

Bouke Stam (1664653)
Scott Mackay (1662769)
16 April 2016

1.2. Doel

Het doel van de deze implementatie is het opslaan van een 2D grid van pixel waarden op de meest efficiente manier, maar tegelijk ook makkelijk te gebruiken is.

1.3. Methoden

Het opslaan van deze grid van pixel waarden is op een groot aantal manieren te realiseren. Hieronder bespreken wij een paar implementaties die door ons onderzocht zijn.

Een van die manieren is alle pixels opslaan in een enkele pointer en daarbij de width en de height van de hele grid op te slaan in twee aparte variablen. Deze methode houdt het gebruik maken van het grid erg simpel omdat het erg basic access heeft.

Een andere methode is een dubbele array aanmaken met de width en height in de grootte van de array verwerkt. Het nadeel hiervan is dat de width en height moeilijker op te vragen zijn. Het voordeel hiervan is dat je gedwongen in twee dimensies moet denken.

Een andere implementatie is het gebruik maken van een std::vector<pixel>. Hierdoor is het makkelijker en sneller om de width en height van een al bestaand grid te veranderen. Het nadeel hiervan is de snelheid van read en write in vergelijking tot een plain array. Arrays zijn O(1) met een hoop acties, en een std::vector<pixel> is in een groter aantal situaties > O(1).

Verder hebben we nog een std::unordered_map<location, pixel>. Deze methode heeft als voordeel dat er zelf bepaald wordt welke locatie bij welke pixel zit. Als een locatie geen data heeft, hoeft het ook niet opgeslagen te worden. Helaas moet wel apart de width en height onthouden worden. Andere nadelen zijn: een onnodige hoeveelheid geheugen gebruik en het over algemeen slomer gebruik dan bij een array.

1.4. Keuze

Uiteindelijk is er gekozen voor de enkele pointer met de width en height variablen erbij. Hiervoor is gekozen omdat deze methode het meest flexibel was, minste memory gebruikte en een van de snelste functies heeft.

1.5. Implementatie

Zoals bij 1.3 al beschreven was, is het geimplementeerd met een pointer met een width en height variable erbij. In de code was in de klas al een width en height opgeslagen in de image zelf. Daardoor was het alleen nodig om de volgende variablen in de klassen van de bijbehorende image te zetten: “`RGB* pixels;`” voor de RGBImageStudent en “`Intensity* pixels;`” voor de IntensityImageStudent.

1.6. Evaluatie

Omdat deze datastructuur gebruikt zal worden in een hoop functies van een van de image klassen, zal deze manier veel op de proef gesteld worden. Daarom bij het testen van alle functies van de RGBImageStudent en de IntensityImageStudent in een meetrapport zal deze manier van opslag op de proef gesteld worden.