Meetrapport image snelheid

1.1. Namen en datum

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

1.2. Doel

Het doel van dit meetrapport is het testen van de snelheid van onze functies en daarbij aantonen dat onze functies snel genoeg zijn voor later gebruik.

1.3. Hypothese

We verwachten dat onze implementaties iets langzamer zijn dan de geleverde RGBImagePrivate en IntensityImagePrivate.

1.4. Werkwijze

Om te testen hoe snel de methodes zijn hebben wel elke methode 10,000 keer uitgevoerd en gemeten hoe lang het duurde. Vervolgens hebben we de totale tijd gedeeld door 10,000 om uit te vinden hoe lang 1 methode gemiddeld duurt.

Dit hebben we gedaan voor de student RGBImage en IntensityImage en dit vergeleken met de private RGBImage en IntensityImage.

1.5. Resultaten

Hieronder zijn de vergelijkingen in snelheid van onze code en de al geleverde code. Daarbij willen wij opmerken dat de set() function geimplementeerd met meer functionaliteit dan de geleverde set() function (wij hebben namelijk een resize gedaan zonder data verlies).

Test	**RGBStudent**	**RGBPrivate**	**IntensityStudent**	**IntensityPrivate**
setPixel(i, value)	0.03 us	0.03 us	0.03 us	0.03 us
getPixel(i)	0.03 us	0.03 us	0.03 us	0.03 us
setPixel(x, y, value)	0.06 us	0.06 us	0.06 us	0.06 us
getPixel(x, y)	0.06 us	0.06 us	0.06 us	0.06 us
RGBImage...(200, 200)	2800 us	2800 us	55 us	55 us
RGBImage…()	2.2 us	1.2 us	2.1 us	1.2 us
set(100, 100) en set(200, 200) afwisselend	1900 us	1700 us	350 us	35 us

1.6. Conclusie

Onze implementatie is op veel vlakken even snel als de geleverde implementatie. Vooral op de cruciale punten (getten en setten van pixels) doen onze methoden niet onder voor de private images.

Alleen bij het maken van een nieuwe image door de constructor met een grootte of de set methode aan te roepen zijn wij langzamer. Dit komt omdat wij de oude pixels behouden en overkopieren naar de vergrootte afbeelding. Wij dachten dat dit de bedoeling was aangezien in de header file stond: “//TODO: resize or create a new pixel storage (Don't forget to delete the old storage)”.

1.7. Evaluatie

Door verschillende snelheden van een computer en onzekerheid van process tijd, is het moeilijk om te zeggen dat de ene functie sneller is dan de andere (bij kleine tijd verschillen). Ondanks dat probleem bleven de tijden van beide implementaties dicht bij elkaar bij veel functies. De tests die we deden bleken dus goed genoeg om te bewijzen dat we snelle functies hebben gemaakt.