1. 1D vs 2D array's

1.1. Namen en datum

Hamza ait Messaoud, 01-01-2018

1.2. Doel

In dit experiment wordt nagegaan welke vorm van dataopslag sneller is. Hierbij wordt een 1-dimensionale array vergeleken met een multidimensionale array. In dit voorbeeld maken we gebruik van een 2-dimensionale array.

1.3. Hypothese

De verwachting is dat een 1-dimensionale array sneller werkt, deze vorm van opslag vergt minder geheugen instructies. Bij een multidimensionale array moet voor elke dimensie het adres opgehaald worden voordat de data opgehaald kan worden.

1.4. Werkwijze

Eerst worden 20 testen uitgevoerd met de 1-dimensionale array en daarna worden dezelfde testen uitgevoerd met de 2-dimensionale array. Vervolgens wordt gemiddelde genomen om de testen te kunnen vergelijken.

Het volgende stuk code is gebruikt om te testen. Om memory allocatie niet mee te nemen in de test is gekozen om van tevoren al de image aan te maken. Maar om de memory allocatie van de array wel mee te nemen wordt de image telkens weer terug gezet op 0 bij 0. Bij het zetten van de afbeelding is gekozen om een copy te maken van een andere afbeelding waardoor er 20 keer een read en 20 keer een write wordt uit gevoerd.

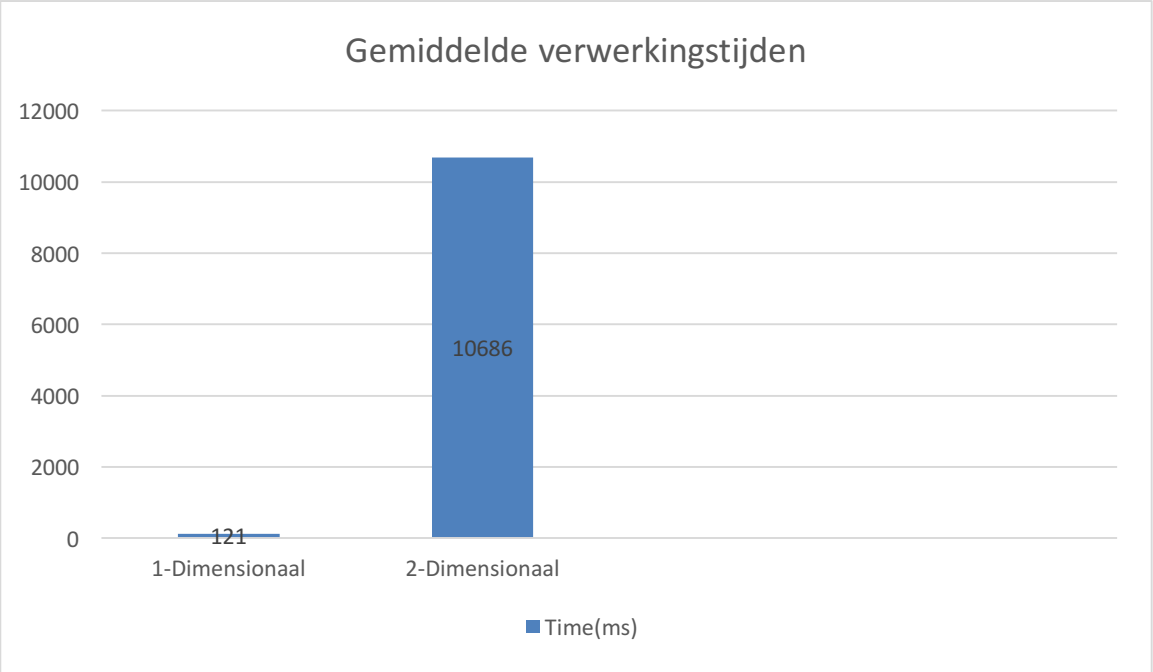
```
RGBImageStudent * copyImage = new RGBImageStudent();

Clock_t startTime, endTime;
for(int i = 0; i < 20; i++) {
	startTime = clock();
	copyImage->set(*image);
	endTime = clock();

	std::cout << "Time: " << endTime - startTime << "\n";

	copyImage->set(0, 0);
}
```

1.5. Resultaten

	1-Demensionaal	2-Demensionaal
	126 µs	10867 µs
	119 µs	11515 µs
	118 µs	10663 µs
	120 µs	10914 µs
	124 µs	11137 µs
	137 µs	10714 µs
	123 µs	10867 µs
	120 µs	10670 µs
	119 µs	10774 µs
	120 µs	10708 µs
	125 µs	10694 µs
	122 µs	10560 µs
	122 µs	10294 µs
	121 µs	10378 µs
	120 µs	10154 µs
	119 µs	10620 µs
	119 µs	10507 µs
	120 µs	10684 µs
	121 µs	10460 µs
	118 µs	10558 µs
Gemiddeld	**121 µs**	**10686 µs**

1.6. Verwerking

Hieronder bevindt zich een grafiek met een gemiddelde opslagtijd van hetzelfde beeld bij de verschillende methoden.



1.7. Conclusie

Uit dit onderzoek kunnen we concluderen dat de 1-dimensionale array efficiënter werkt dan de 2-dimensionale array.

1.8. Evaluatie

Onze hypothese klopt. De hypothese was gemaakt op basis van de beschikbare kennis van assembly. Een 1-dimensionale array maakt minder geheugen aanvragen dan een 2-dimensionale array. Van een 2-dimensionale array bevat de eerste array alleen pointers naar de tweede array. In de tweede array zijn de pixels en informatie opgeslagen. Dus als een pixel veranderd moet worden kost dat meer instructies. Daarnaast kan er bij een 1-dimensionale array meer snelheid geboekt worden met het kopiëren van data.