Gerrit van Os & Bryan Campagne

[Trek de aandacht van uw lezer met een veelzeggend citaat uit het document of gebruik deze ruimte om een belangrijk punt te benadrukken. Sleep dit tekstvak als u het ergens anders op de pagina wilt plaatsen.]

Meetrapport snelheid

Imageshell en grayscale

Inhoud

**Geen inhoudsopgavegegevens gevonden.**

# 1. Doel

In dit onderzoek zal er gekeken worden naar de snelheid van de imageshell en het gekozen grayscale algoritme.In dit ondezoek zullen een aantal tests gedaan worden die de door ons geïmplementeerde image shell en grayscale algoritme gaan vergelijken met de default implementatie.

# 2. Hypothese nog uitbreiden

We verwachten dat onze implementatie van de image shell en gray scaling sneller zijn dan die van de standaard implementatie.

# 3. Werkwijze

Er zullen een aantal testen uitgevoerd worden, zowel voor de image shell als de grayscale hieronder zullen deze per onderdeel beschreven worden.

## 3.1 Algemeen

Voor het timen van de code zal er gebruik gemaakt worden van een std::chrono::high\_resolution\_clock voorafgaand aan het runnen van de samplesizes zal de now() methode aangeroepen worden om de huidige tijd op te vragen, na het runnen van de for-loop zal nogmaald de now() functie aangeroepen worden het verschil van deze 2 tijden is de verstreken tijd.

Alle tests zijn gerunt vanuit visual studio in debug build mode.

Alle test zijn uitgevoerd op een HP Pavilion Power Laptop 15-cb0xx met de volgende specificaties:

* **Processor:** i7-7700hq
* **Geheugen:** 16GB DDR4 2400MHz
* **Opslag:** 256gb nvme ssd

## 3.2 Grayscale

Voor het testen van het grayscale algoritme zal er een vergelijking gemaakt worden met de default implementatie. Er zal getest worden met verschillende sample sizes: 10,100,1000 en 10.000. Om alleen de grayscale functie te testen zal er een for-loop om de PreProcessingStep1 geplaatst worden. Doormiddel van deze for-loop kan het aantal keer uitvoeren veranderd worden. Hieronder het stuk code wat gebruikt is.

auto startGrayScale= std::chrono::high\_resolution\_clock::now();

for (int i = 0; i <= 10; i++) {

if (!executor->executePreProcessingStep1(false)) {

std::cout << "Pre-processing step 1 failed!" << std::endl;

return false;

}

}

auto finishGrayScale = std::chrono::high\_resolution\_clock::now();

elapsedGrayScale = finishGrayScale - startGrayScale;

In dit stukje code is de i<=10 bepalend voor het aantal samples.

## 3.3 Imageshell:

Voor het testen van de imageshell gaan we bijna op dezelfde manier te werkt, echter omdat de imageshel gedurende het gehele programma gebruikt wordt zal er hier gekeken worden naar de runtime van het gehele programma in plaats van naar 1 functie. Vanwege de lengte van deze tests zullen de sample sizes iets anders liggen namelijk: 10,50,100 en 250 de manier van de code timen gaat op dezelfde manier alleen nu zal de for-loop de volledige main beslaan en dus het gehele programma timen. Hieronder is te zien hoe we dit in code geïmplementeerd hebben:

int main(int argc, char \* argv[]) {

auto startProgram = std::chrono::high\_resolution\_clock::now();

//ImageFactory::setImplementation(ImageFactory::DEFAULT);

ImageFactory::setImplementation(ImageFactory::STUDENT);

for (int i = 0; i <= 250; i++) {

--------- code van de originele main

}

auto finishProgram = std::chrono::high\_resolution\_clock::now();

std::chrono::duration<double> elapsedProgram = finishProgram - startProgram;

std::cout << "time elapsed total program: " << elapsedProgram.count() << "\n";

system("pause");

return 1;

}

# 4. Resultaten

## 4.1 Grayscale

Hieronder een tabel van de tijden die gemeten zijn tijdens het testen van de verschillende grayscale algoritmes. Bovenin de tabel is de samplesize weergegeven met daaronder de gemeten tijd in seconden.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Grayscale snelheid | 10 | 100 | 1000 | 10000 |
| Student implementatie | 0.239 | 2.22 | 12.5 | 117.5 |
| Default implementatie | 0.39 | 2.9 | 23.9 | 235.7 |
| verschil | 0.151 | 0.68 | 11.4 | 118.2 |

Tevens nog een tabel met de gemiddelde tijd per functie deze is uitgedrukt in milliseconde en gebaseerd op de deling van het resultaat en de samplesize.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Grayscale snelheid gem | 10 | 100 | 1000 | 10000 |
| Student implementatie | 23.9 | 22.2 | 12.5 | 11.75 |
| Default implementatie | 39 | 29 | 23.9 | 23.57 |
| verschil | 15.1 | 6.8 | 11.4 | 11.82 |

## 4.2 Imageshell

Voor de imageshell zijn hieronder de resultaten weergegeven in eenzelfde soort tabel eerst de totaaltijd per run in seconden daarna de gemiddelde tijd in milliseconde.