

1. Memory efficiency test Week 1

1.1. Namen en datum

Namen: Martijn van der Struijk en Tim Hasselaar.

Datum: 16-04-2015

1.2. Doel

Het doel van dit experiment is om duidelijkheid te krijgen over het verschil tussen het geheugen gebruik van de default code en de student code. Hierdoor kunnen wij uiteindelijk bepalen of onze code efficiënt is (er vanuit gaande dat de default code goed geoptimaliseerd is).

1.3. Hypothese

Wij denken dat we voor onze code een vrij efficiënte manier hebben gerealiseerd om alle data op te slaan. We verwachten niet meer dan een verschil van 5% tussen het geheugengebruik van de default code en de student code.

1.4. Werkwijze


In de main.cpp van het project gaan wij 200 extra RGBImages aanmaken, met een grootte van 1000x1000. Wij gebruiken zo veel images zodat kleine verschillen erg uitvergroot zullen worden. Vervolgens zullen wij via het taakbeheer kijken hoeveel werkgeheugen het programma gebruikt. Door dit te vergelijken bij de default en de student code kunnen wij de efficiëntie bepalen.

1.5. Resultaten

Als eerst hebben wij 200 extra RGBImages aangemaakt volgens de default code:

```
int main(int argc, char * argv[]) {  
  
    ImageFactory::setImplementation(ImageFactory::DEFAULT);  
    //ImageFactory::setImplementation(ImageFactory::STUDENT);  
  
    ImageIO::debugFolder = "C:\\TestSetVision\\FaceMinMin";  
    ImageIO::isInDebugMode = true; //If set to false the ImageIO class will skip any image save function calls  
  
    //De memory efficiency test  
    for (int i = 0; i < 200; i++){  
        RGBImage * test = ImageFactory::newRGBImage(1000, 1000);  
    }  
}
```


Dit resulteerde in een geheugen gebruik van 574.9 MB.

Name	Status	19% CPU	53% Memory
Apps (8)			
▶  ExternalDLL.exe (32 bit)		0%	574,9 MB

Vervolgens voerden wij deze test uit met de student code:

```
int main(int argc, char * argv[]) {  
  
    //ImageFactory::setImplementation(ImageFactory::DEFAULT);  
    ImageFactory::setImplementation(ImageFactory::STUDENT);  
  
    ImageIO::debugFolder = "C:\\TestSetVision\\FaceMinMin";  
    ImageIO::isInDebugMode = true; //If set to false the ImageIO class will skip any image save function calls  
  
    //De memory efficiency test  
    for (int i = 0; i < 200; i++){  
        RGBImage * test = ImageFactory::newRGBImage(1000, 1000);  
    }  
}
```

Dit resulteerde in een geheugengebruik van 574.9 MB.

Name	Status	0% CPU	53% Memory
Apps (9)			
▶  ExternalDLL.exe (32 bit)		0%	574,9 MB

1.6. Verwerking

Het extra geheugen dat onze code gebruikt is te berekenen met de formule:

$$((\text{STUDENT GEHEUGEN}) / (\text{DEFAULT GEHEUGEN}) - 1) * 100\%$$

1.7. Conclusie

Als we deze formule invullen krijgen we:

$$(574.9/574.9 - 1) * 100\% = (1-1) * 100\% = 0\%$$

Hier uit kunnen wij concluderen dat onze eigen code 0% meer geheugen gebruikt dan de default code, en dat onze hypothese klopt.

1.8. Evaluatie

Wij denken dat wij voor onze eigen code dezelfde opslag methode hebben gekozen die bij de default code word gebruikt. Het bepalen van de efficiëntie is dus gelukt. Onze student code gebruikt precies even veel geheugen als de default code.