1. Implementatieplan titel

1.1. Namen en datum

Edwin Koek & Jacob Visser 2-6-2015

1.2. Doel

Het doel is om een image te schalen naar <=40000 pixels zonder dat het ten koste gaat van de kwaliteit van de image.

1.3. Methoden

Ten eerste zijn er 2 methoden om tot een nieuwe image te komen.
- Forward mapping
- Backward mapping

Wanneer forward mapping wordt toegepast wordt voor elke oude pixel een nieuwe coördinaat berekend. Bij backward mapping wordt voor elke nieuwe pixel de oude coördinaat berekend.

Forward mapping heeft een paar grote nadelen.
-Sommige nieuwe coördinaten vallen buiten de nieuwe image.
-Berekende coördinaten zijn niet altijd gehele getallen.
-Niet voor elke pixel in de oude image in een pixel in de nieuwe image

Het 3rde nadeel is het grootste nadeel. Wanneer er pixels zijn waar geen nieuwe pixel voor is dan komen er gaten in de image.

Nadat de coördinaten berekend zijn moet de kleur van de nieuwe of oude pixel berekend worden. Dit kan op meerdere manieren:
- 0th order interpolatie (nearest-neighbor)
- 1st order interpolatie (bilinear)
- 3rd order interpolatie(bicubic)

Bij nearest-neighbor wordt simpelweg afgerond naar de kleur van de dichtstbijzijnde pixel. Deze methode is snel, maar geeft een blokkerig resultaat. Bij bilinear interpolatie wordt een gewogen gemiddelde kleur berekend van de 4 omliggende pixels. Deze methode is minder snel dan nearest-neighbor, maar geeft wel een minder blokkerig resultaat. Tot slot is er ook nog bicubic interpolatie. Deze is vrijwel hetzelfde als bilinear, maar dan wordt een gewogen gemiddelde genomen van de omliggende 16 pixels. Deze methode is weer een stuk trager dan bilinear, maar geeft een nog scherper resultaat.

1.4. Keuze

Wij hebben gekozen om backward mapping met bilinear interpolatie toe te passen. Backword mapping is gekozen om alle nadelen van forward mapping te vermijden. Bilinear interpolatie is gekozen omdat het vrij snel een goed resultaat levert. Nearest-neighbor is niet gekozen omdat de resulterende images te blokkerig zijn. Daarnaast is de tijd winst zo klein dat het te verwaarlozen is. Bicubic interpolatie is niet gekozen omdat de extra tijd die het systeem nodig heeft om dit toe te passen het naar onze mening niet waard is. Daarnaast is het vrij lastig om te implementeren.

1.5. Implementatie

```
IntensityImage * StudentPreProcessing::stepScaleImage(const IntensityImage &image) const {
    IntensityImageStudent* IM = new IntensityImageStudent(image.getWidth(), image.getHeight());
    // sla het schalen van de image over als de image al van het juist formaat is
    if (image.getWidth() * image.getHeight() <= 40000){
        for (int y = 0; y < image.getHeight(); ++y){
            for (int x = 0; x < image.getWidth(); ++x){
                IM->setPixel(x, y, image.getPixel(x, y));
            }
        }
        return IM;
    }
    // Bereken waarmee de X en Y as van de image vermenigvuldigd moet worden
    // om de aspect ratio te behouden
    int pixelCount = image.getHeight() * image.getWidth();
    float multiplier = static_cast<float>(40000) / pixelCount;
    multiplier = sqrt(multiplier);
    // Schaal de image
    IM = new IntensityImageStudent(image.getWidth() * multiplier, image.getHeight() * multiplier);
    // Pas bilinear interpolatie toe om de intensiteit van de pixels in de nieuwe image
    // te bepalen.
    Intensity newIntensity = 0;
    for (int y = 0; y < IM->getHeight(); ++y){
        for (int x = 0; x < IM->getWidth(); ++x){
            float oldX = x / multiplier;
            float oldY = y / multiplier;
            float dX = 1 - (oldX - floor(oldX));
            float dY = 1 - (oldY - floor(oldY));
            newIntensity = (dX * dY) * image.getPixel(floor(oldX), floor(oldY));
            newIntensity += ((1 - dX) * dY) * image.getPixel(floor(oldX) + 1, floor(oldY));
            newIntensity += (dX * (1 - dY)) * image.getPixel(floor(oldX), floor(oldY) + 1);
            newIntensity += ((1 - dX) * (1 - dY)) * image.getPixel(floor(oldX) + 1, floor(oldY) + 1);
            IM->setPixel(x, y, newIntensity);
        }
    }
    return IM;
}
```

1.6. Evaluatie

Er wordt getest of de images daadwerkelijk goed gescaled worden naar <=40000 pixels. Daarnaast wordt ook gekeken naar de efficiëntie van de gekozen methode. Dit wordt gedaan door het te vergelijken met een nearest-neighbor implementatie en de default implementatie.