Institut für Informatik
Henning Wenke M.Sc.
Sascha Kolodzey B.Sc.
Nils Vollmer B.Sc.

Universität Osnabrück
http://www-lehre.inf.uos.de/~pa/

Übung: Parallele Algorithmen mit OpenCL

Sommersemester 2013

Blatt 8

Laden Sie sich zur Bearbeitung der Aufgaben das bereitgestellte Framework unter folgendem Link http://www-lehre.inf.uos.de/~pa/Uebungen/Blatt8/PA_Blatt8.zip herunter und installieren Sie es in einer Entwicklungsumgebung. Um die jeweilige Aufgabe zu starten, müssen Sie lediglich die zu startende Klasse in der `Main` Klasse angeben.

Aufgabe 8.1 CONVOLUTION: Edge detection (50 Punkte)

In dieser Aufgabe sollen Sie den in der Übung vorgestellten CONVOLUTION Algorithmus implementieren und mit Hilfe des Sobel-Operators als Convolution Maske Kanten innerhalb eines Bildes erkennen. Der Sobel-Operator ist eine Faltung (Convolution), die aus dem Originalbild ein Gradienten-Bild erzeugt und dadurch hohe Frequenzen über Grauwerte darstellt. Die Faltungen werden jeweils für die X - und Y - Dimension folgendermaßen aufgestellt:

$$G_X = S_X \cdot A, \, mit \, S_X = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

und

$$G_Y = S_Y \cdot A, \, mit \, S_Y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Der letztendliche Grauwert G berechnet sich aus dem arithmetischen Mittel des Betragsvektors $V = \sqrt{G_X \cdot G_X + G_Y \cdot G_Y}$.

Schauen Sie sich zur Bearbeitung dieser Aufgabe die Klasse `EdgeDetection.java` an. Alle Stellen, an denen Sie Änderungen vornehmen müssen, sind im Code sichtbar gemacht. Den Kernel für diese Aufgabe müssen Sie von Grund auf schreiben. Überlegen Sie sich, welche Argumente Sie benötigen. Die benötigten Bilddaten werden Ihnen bereits als FloatBuffer `pixel` im Format `RGBA (float4)` bereitgestellt. Starten Sie für jeden Bildpunkt einen Thread und berechnen Sie mit Hilfe des Sobel-Operators den resultierenden Grauwert. Schreiben Sie das Ergebnis in den dafür vorgesehenen Buffer `edgedImage`. Der Buffer `edgedImage` liegt ebenfalls im Format RGBA vor. Schreiben Sie deshalb einfach in jede Komponente den berechneten Wert. Da sich die Convolution Maske auch zur Laufzeit nicht ändern soll, soll die Maske direkt als ein mit `constant` deklariertem Array hart in das OpenCL Program geschrieben werden. Globale konstante Arrays innerhalb eines Programs legen Sie wie folgt an:

```
constant float a[5] = {1,2,3,4,5};
```

Aufgabe 8.2 CONVOLUTION: Gaussian Blur (50 Punkte)

Als zweites Beispiel sollen Sie einen Gaussian Blur Filter implementieren. Der Gaussian Blur Filter ist weit bekannt aus Grafikprogrammen, um Noise oder Details eines Bildes zu reduzieren. Wie der Name schon vermuten lässt, basiert der Filter auf der Gauß-Funktion. Für zwei Dimensionen ist der Filter wie folgt definiert:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Die Funktion berechnet das Gewicht, mit welchem der Pixel, mit dem Abstand (x, y) zum betrachteten Pixel $(0, 0)$, in die Berechnung einfließt. Für die Berechnung des neuen Wertes werden alle Nachbar Pixel mit diesem Gewicht multipliziert und aufsummiert. Das Ergebnis ist ein gewichteter Mittelwert aller Nachbar Pixel. Schauen Sie sich für diese Aufgabe die Java Klasse `Blur.java` an. Auch hier sind an allen Stellen im Code Markierungen, die abzuarbeiten sind. Die Strukturen und Daten sind im Wesentlichen so angelegt wie in Aufgabe 8.1. In dieser Aufgabe soll die Convolution Maske, anders als in Aufgabe 8.1, zur Laufzeit über User-Input verändert werden können. Es reicht daher nicht, eine vorher berechnete Maske hart in das Program zu schreiben. Übergeben Sie die Maske an ein mit `constant` deklariertes Argument im Kernel. Benutzen Sie hierfür den vorgesehenen Buffer `convolutionMask`. Der User soll zwei Parameter über die Pfeiltasten variieren können:

- Filtergröße, ↑ größer und ↓ kleiner
- "Einflussstärke" der Nachbarn (σ Standardabweichung), ← kleiner und → größer

Die einzelnen Parameter werden innerhalb der Klasse `Blur.java` durch die private Klasse `Settings` verwaltet. Um das Abfangen des Inputs müssen Sie sich

nicht kümmern. In der Klasse `Blur.java` stehen bereits Methoden zur Verfügung, die beim Drücken der jeweiligen Taste aufgerufen werden. Außerdem wird nach jeder Änderung durch den User die Methode `onSettingsChanged` aufgerufen. In dieser Methode müssen Sie eventuell neue Kernel Argumente setzen und den Buffer der Convolution Maske neu erstellen. Stellen Sie sicher, in dieser Methode keine Memory Leaks zu erstellen. Zur Generierung der Convolution Maske müssen Sie die Methode

`getGaussianBlurConvMask(int size, double sigma)`

implementieren. Beachten Sie, dass wir nur an symmetrischen Masken interessiert sind und deshalb nur eine Dimensionsgröße `(size)` angegeben werden muss. Nutzen Sie für die Berechnung der Gewichte die oben angegebene Funktion.

Hinweis:
Convolution Masken haben in der Regel eine ungerade Anzahl an Elementen in ihren Dimensionen. Das macht Sinn, weil bei geraden Anzahlen immer ein Element auf einer Seite zu wenig wäre. Es mag Anwendungen geben, wo dies gewollt sein kann aber normalerweise ist eine symmetrische Berechnung gewünscht.