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 9

## Aufgabe 8.1 Raytracing (50 Punkte)

In dieser Aufgabe sollen Sie sich über die Implementierung einer Komponente eines Raytracers Gedanken machen. Ein Raytracer generiert auf Basis einer Szene ein zweidimensionales Bild. Anders als in der Rastergrafik, verfolgt ein Raytracer den Ansatz, Strahlen in die Szene zu schießen und diese an ihren Schnittpunkten mit Objekte der Szene auszuwerten. Auf Basis der Materialeigenschaften der getroffenen Objekte, wird über die weitere Verarbeitung entschieden. Sollte ein Strahl z.B. auf ein Stück Glas treffen, werden neue Strahlen erzeugt und entsprechend der Reflektion/Refraktion weiter durch die Szene verfolgt. Da dies aber in der Regel nicht für alle Strahlen geschieht, dünnt sich mit jedem Iterationsschritt die Anzahl der zu verfolgenden Strahlen weiter aus, bis schließlich alle Strahlen terminiert sind. Eine naive Implementation wäre, für jeden Bildpunkt einen Thread zu erzeugen, und mit einer While-Schleife solange den Strahl zu verfolgen, bis dieser abgearbeitet ist. Man überlege sich an dieser Stelle das grundlegende Problem dieses Ansatzes. Der Pseudocode einer besseren Implementierung könnte folgendermaßen aussehen:

```
while(running)
 rayCount = width * height
 rays = initRays()
 hits = initHits()
 image = black_image
 while(rayCount > 0)
  in parallel do computeRayHits(rays, hits)
  in parallel do computeColorContribution(image, hits)
  in parallel do computeNewRays(rays, hits)
  in parallel do arrangeNewRays(rays, rayCount)
 end
 showImage(image)
end
```

Sie sollen in dieser Aufgabe die Funktion (Kernel) `arrangeNewRays` implementieren. Als Eingabe bekommt die Funktion die Ausgabe der Funktion `computeNewRays`. Die Funktion `computeNewRays` berechnet für alle aktuellen Rays, ob diese in der nächsten Iteration weiter verfolgt werden sollen, weil sie eventuell auf eine spiegelnde Oberfläche getroffen sind. Zur Vereinfachung wird angenommen, dass ein Ray maximal einen weiteren Ray erzeugen kann. Um weitere Schritte effektiv durchführen zu können, sollen alle Speicheradressen der Rays für die nächste Iteration, nach Aufruf der Funktion `arrangeNewRays`, konsekutiv im Speicher vorliegen.
Zur Anschauung dient folgendes Beispiel (`Exists` und `MemoryIndex` seien Arrays):

```
Exists      : 1 1 1 1 1 1 1 1 1 <- initRays
MemoryIndex: 0 1 2 3 4 5 6 7 8 <- initMemoryIndex

Exists      : 1 1 0 1 1 0 1 0 1 <- computeNewRays()
MemoryIndex: 0 1 3 4 6 8 X X X <- arrangeNewRays()
```

`Exists` speichert, ob der Ray an diesem Index weiterverfolgt werden soll. In `MemoryIndex` werden dann konsekutiv die Speicheradressen hinterlegt. Überlegen Sie sich eine Möglichkeit, das Problem parallel mit den Ihnen bekannten Mitteln aus der gesamten Vorlesung zu lösen. Wenn Sie letztendlich keine funktionierende Implementierung schaffen, sollten Sie dennoch alle Ideen und Ansätze schriftlich festhalten und in Ihrem Testat ihrem Tutor präsentieren.
Zur Vereinfachung liegt ein Template für den Aufbau des Programms bereit:
http://www-lehre.inf.uos.de/~pa/Uebungen/Blatt9/PA_Blatt9.zip
**Hinweis:** Für funktionierende Lösungen, die aber keinen Sinn auf paralleler Hardware machen, werden eventuell keine Punkte vergeben.

### Aufgabe 8.2 Indexvergabe dynamischer Daten (35 Punkte)

Überlegen Sie sich Lösungsmöglichkeiten zu folgender Problemstellung:
Aus einem sich über die Zeit ändernden Datensatz, sollen Objekte erkannt werden. Dazu wird für jeden Eintrag der Daten ein Work-Item gestartet. Ein Work-Item kann anhand der ihm zugewiesenen Daten erkennen, ob ein Objekt vorliegt und diesem dann einen eindeutigen Index zuweisen. Welche Möglichkeiten gibt es, den Index lückenlos aufsteigen zu lassen und/oder ihn reproduzierbar zu verteilen. Schreiben Sie Ihre Ideen auf und erklären Sie diese Ihrem Tutor. Bewerten Sie außerdem die von Ihnen gefundenen Lösungen nach den in der Vorlesung aufgeführten Bewertungen für parallele Algorithmen.

### Aufgabe 8.3 Bewertung & Eigenschaften paralleler Algorithmen (15 Punkte)

Bewerten Sie die Implementierungen der N-Body Simulation, REDUCTION und CONVOLUTION der letzen Übungsblätter und beide in der Vorlesung vorgestellten SCAN Varianten nach den in der Vorlesung aufgeführten Bewertungen für parallele Algorithmen.