Hochparallele Simulationsrechnungen mit CUDA und OpenCL

3. Übung

Das Spiel des Lebens ist ein vom Mathematiker John Conway entworfenes System, basierend auf einem zweidimensionalen zellularen Automaten. Das Spielfeld ist in Zeilen und Spalten unterteilt. Jede Zelle des Spielfeldes kann einen von zwei Zustanden einnehmen kann, welche als lebendig und tot bezeichnet werden. Die nächste Generation einer Konfiguration ergibt sich durch einfache Regeln.

- Eine tote Zelle mit genau drei lebenden Nachbarn wird in der Folgegeneration neu geboren.
- Eine lebende Zelle mit zwei oder drei lebenden Nachbarn bleibt in der Folgegeneration lebend.
- Lebende Zellen mit einer anderen Anzahl an lebenden Nachbarn sterben in der Folgegeneration.

Weiterhin ist es problemlos möglich andere Regeln zu definieren. Conways Welt ist eine 23/3-Welt. (Eine Zelle bleibt bei 2 oder 3 lebendigen Nachbarn lebendig. Eine Zelle wird neu geboren bei exakt 3 lebendigen Nachbarn.)

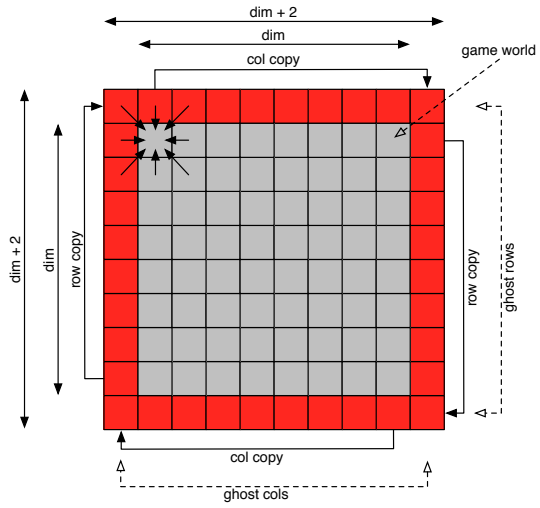
Aufgaben:

a) Implementieren Sie mit Hilfe des vorgegebenen Codegerüstes eine sequentielle Variante des Game of Life. Verwenden Sie hierbei die klassischen Regeln von Conway und implementieren Sie eine Welt deren Ränder miteinander verbunden sind.

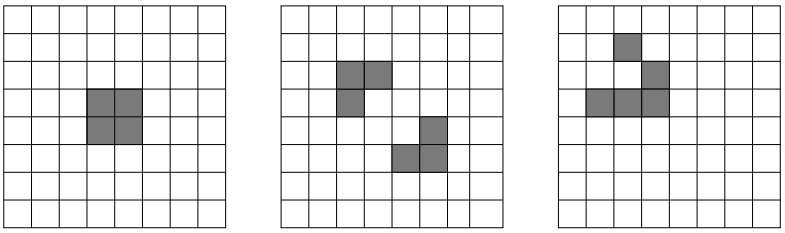
Die Dimension des quadratischen Spielfeldes sollte dynamisch veränderbar sein. Nutzen Sie hierzu die Funktion `malloc(...)` um einen linear adressierbaren Speicherbereich festzulegen und überlegen Sie sich eine sinnvolle Abbildung der Matrizen in diesen Speicherbereich.

Hinweise:

- Nutzen Sie folgende Abbildung zur Realisierung der verbundenen Ränder unter der Nutzung von Schatten-Reihen und -Spalten.



b) Überprüfen Sie Ihre Implementierung anhand der unten abgebildeten Objekte. Diese sollten statisch, oszillierend und bewegt sein.



Statischer Block | Oszillator | Gleiter

Bei einer korrekten Implementierung mit einem Zufallsbild entstehen weder Chaos noch eine leere Welt. Überprüfern Sie das korrekte Verhalten an den Rändern mit einem Gleiter. Bei korrekter Implementierung sollte dieser beim übertreten eines Randes auf der entgegengesetzten Seite wieder erscheinen.

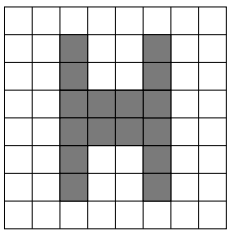
Hinweis:

- Sie können mit der vorgegebenen Funktion `createPPM(...)` eine PPM-Datei des Spielfeldes generieren und sich so den Endzustand ansehen und das Verhalten überprüfen.

c) Erweitern Sie Ihren Algorithmus, so dass die Berechnung der nächsten Generation des Problems auf der GPU erfolgt. Verteilen Sie die Arbeit hierbei auf die Multiprozessoren der GPU.

Hausaufgabe:

- Implementieren Sie die Regeln einer 1357/1357-Welt. Diese liefern eine Kopierwelt, welche ihren Inhalt vervielfältig. Initialisieren Sie ein Feld der Größe 1024 x 1024 und initialisieren Sie es in der Mitte mit einem H.



Buchstabe H für Kopierwelt

- Lassen Sie Ihren Algorithmus laufen und überprüfen Sie die Korrektheit Ihrer Regeln. Das H sollte immer wieder vervielfaltigt werden.
- Bestimmen Sie weiterhin den Speedup, welcher durch die Berechnung auf der GPU entsteht und ermitteln Sie, ab welcher Anzahl an berechneten Generationen die GPU einen Laufzeitvorteil gegenüber der CPU hat. Stellen Sie den erreichten Speedup für verschiedene Problemgrößen grafisch dar (z.B. mit Gnuplot).