Modellierung und Simulation 2019/2020
Conway's Game of Life

Louis Donath, Dario Klepoch

Potsdam University

Zusammenfassung. Conway's Game of Life ist ein Automat... Hier kommt eine kurze Zusammenfassung des Projektes rein

1 Einfuehrung

'Conways game of life' (im folgenden: GoL)

2 Erklaerung unserer Implementierung

3 Implementierung

Bei der Betrachtung der Implementierung werden wir ueber die Implementierung der Randbedingungen sprechen. Weil wir mit unseren ersten Implementierung mit einer unzureichenden Performance hatten, werden wir im zweiten Teil ueber Performanceverbesserungen sprechen.

3.1 Randbedingungen

In unserer Implementierung des GoL haben wir drei unterschiedliche Randbedingungen implementiert. Eine Randbedingung sagt aus wie sich das Spiel verhält, wenn auf Zellen ausserhalb des Spielbrettes zugegriffen wird.
Absorbierende Randbedingung bedeutet das jede Zelle die ausserhalb des Spielbrettes ist als tot angenommen wird.
Wir haben die **periodische** und **reflektierende Randbedingungen** implementiert wie man den Schemata in Tabelle 1 entnehmen kann. Auf der linken Seite ist die periodische Randbedingung gezeigt und auf der rechten Seite die reflektierende. In der Mitte ist das eigentlich Spielbrett mit den Zahlen 1..9 dargestellt. Darum herum sind (gespiegelte) Kopien des Spielbrettes gezeigt. Die Zahlen $1'..9'$ symbolisieren Referenzen zu dem Spielbrett in der Mitte. Daher wenn die Zelle 1 lebt, leben auch alle Zellen 1'. Wenn sich nun eine Zelle auf den Rand zu bewegt wird sie bei der periodischen Randbedingung am Anfang des Spielbrettes neu erscheinen. Bei der reflektierende hingegen in die gleiche Zelle gespiegelt.

Tabelle 1. Darstellung der Randbedingungen

1' 2' 3'	1' 2' 3'	1' 2' 3'	9' 8' 7'	7' 8' 9'	7' 8' 9'
4' 5' 6'	4' 5' 6'	4' 5' 6'	6' 5' 4'	4' 5' 6'	4' 5' 6'
7' 8' 9'	7' 8' 9'	7' 8' 9'	3' 2' 1'	1' 2' 3'	3' 2' 1'
1' 2' 3'	**1 2 3**	1' 2' 3'	3' 2' 1'	**1 2 3**	3' 2' 1'
4' 5' 6'	**4 5 6**	4' 5' 6'	6' 5' 4'	**4 5 6**	6' 5' 4'
7' 8' 9'	**7 8 9**	7' 8' 9'	9' 8' 7'	**7 8 9**	9' 8' 7'
1' 2' 3'	1' 2' 3'	1' 2' 3'	9' 8' 7'	7' 8' 9'	9' 8' 7'
4' 5' 6'	4' 5' 6'	4' 5' 6'	6' 5' 4'	4' 5' 6'	6' 5' 4'
7' 8' 9'	7' 8' 9'	7' 8' 9'	3' 2' 1'	1' 2' 3'	3' 2' 1'

3.2 Performance

Nachbarn ermitteln

Partielle Updates Nach der ersten Implementierung des GoL war die Performance fuer groesserer Spielfelder nicht ausreichend. Den grössten Einfluss dabei hatte das Zeichnen von den Zellen des Spiels. Daher wird bei unsere Implementierung bei der Berechnung einer neuen Generation eine Liste erstellt mit den Zellen welche sich, im Vergleich zur letzten Generation, verändert haben. Nur die Zellen welche sich in dieser Liste befinden werden neu gezeichnet und alle anderen Zellen bleiben gleich.

Numpyarray anstatt Pythonlist Um die Performance noch weiter zu steigern wurde fuer die Representation des Spielbrettes ein Numpyarray anstelle eines Pythonarrays gewaehlt. Ausserdem ist dieses Array ein Array von boolean. Daher wenn eine Zelle lebendig ist, ist der Wert True (1) sonst False (0).

Weitere Performancesteigerung sind durch sehr viele unterschiedliche Veraenderungen moeglich. Eine sehr grossen Performancesteigerung ist dadurch moeglich einen effizientere Datenstruktur als ein (numpy-)array zu verwenden. In verschiedenen anderen Implementierungen des GoL wird hierfuer ein Quadtree benutzt. Ein Quadtree wird meistens dafuer verwendet effizient 2-dimensionale Daten zu speichern [1]. Da das GoL auch 2D-Daten sind ist es ein perfekter Anwendungsbereich fuer einen Quadtree. Mit 'Haslife' wurde das GoL auf diese Weise implementiert [2].
Um die Performance noch weiter zu steigern ist es moeglich den Quadtree parallel aufzubauen. Hier kann entweder die CPU oder auch die GPU benutzt werden. In [3] wurden lineare Quadtrees verwendet um einen Quadtree vollstaendig auf der GPU aufzubauen.

Literatur

1. `https://www.geeksforgeeks.org/quad-tree/`, letzter Zugriff: 27.3.2020

2. `https://en.wikipedia.org/wiki/Hashlife`, letzter Zugriff: 27.3.2020
3. Dupuy, Jonathan & Iehl, Jean-Claude & Poulin, Pierre. (2018). Quadtrees on the GPU. 10.1201/9781351052108-12.