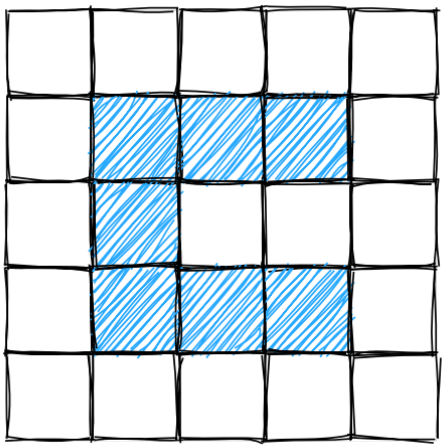
31.5.2018

Cells

Eine Conway's Game of Life Implementation

Florian Fechner
F_FECH03@UNI-MUENSTER.DE



Vorwort

Bei dem folgenden Dokument handelt es sich um eine deutsche Dokumentation der Software „Cells“, welche für die Code Competition von it-talents.de im Mai 2018 erstellt wurde.

Inhaltsverzeichnis

Verwendete Werkzeuge und Softwarepakete

Bei der Implementierung von „Cells“ wurden eine Reihe von Frameworks und Softwarepaketen verwendet. Die Basis hierbei bildet das Softwarepaket Node.js sowie die beiden Paket-Management-Services NPM und Bower.

NPM

Durch den Paket-Management-Service NPM werden Abhängigkeiten installiert, die sich über das Node.js Modulsystem integrieren lassen. Eine zentrale Abhängigkeit stellt hierbei das Modul Electron dar. Durch Electron ist es möglich eine Desktop Applikation zu erstellen, welche intern für die Darstellung der Benutzeroberfläche den Webbrowser Chromium verwendet. Hierdurch wird es für den Ersteller einer Desktop Applikation möglich die komplette Darstellungslogik seiner Applikation in HTML, CSS und JavaScript zu implementieren. Des Weiteren wurde das NPM-Paket rough.js verwendet, um SVG-Elementen den Charme von selbstgezeichneten Bildern zu verleihen. Für effizientere Operationen auf mehrdimensionalen Arrays in JavaScript wurde das NPM-Paket ndarray sowie dessen Untermodul zeros verwendet.

Bower

Durch den Paket-Management-Service Bower lassen sich sogenannte Web Components installieren. Bei Web Components handelt es sich um Elemente einer Website, welche bestimmte W3C-Standards erfüllt und dadurch ein gekapseltes System bilden. Hierdurch wird die technische Verschuldung bei der Verwendung unterschiedlicher Elemente möglichst klein gehalten. Nähere Informationen lassen sich auf https://www.webcomponents.org/ nachlesen. Als Bibliothek zur praktischeren Erstellung von Web Components wurde Polymer verwendet. Des Weiteren wurden einige Web Components der „Paper Elements“ Collection und der der „Iron Elements“ Collection verwendet. Die „Paper Elements“ Collection enthält visuelle Web Components, welche die von Google entworfene Designrichtlinien von Material Design realisieren. Bei der „Iron Elements“ Collection handelt es sich um eine Sammlung von Basis-Elementen für den grundlegenden Aufbau einer Website.

Electron

Die vom W3C definierten Web-Standards bilden in meinen Augen eines der besten visuellen Darstellungssysteme. Die Programmiersprache JavaScript erfreut sich außerdem großer Beliebtheit in der Entwickler Community, weshalb es eine überwältigende Fülle an Bibliotheken für eben diese Sprache gibt. Deshalb sehe ich Electron als eine gute Wahl an, wenn der Entwickler das Ziel hat eine Desktopapplikation zu implementieren, wenn sie nicht grade ein dreidimensionales Videospiel oder eine komplexe Simulation werden soll. In solchen Fällen kann die interpretierte Sprachnatur von JavaScript nämlich ein Problem darstellen. Eine Implementierung von Conway’s Game of Life sehe ich allerdings nicht als solch einen kritischen Fall an, wenn der Spieler nicht unbedingt wünscht auf ein 1000x1000 Feld zu spielen. Solch ein Spieler kann sich ja mit eines der anderen Einsendungen für die Coding Challange vergnügen, die mehr auf Performance und weniger auf Benutzeroberflächengestaltung ausgelegt ist.

Technische Durchführung

Zeitintervall

In der Applikation wird genau ein Intervall in der Wurzelkomponente gestartet, welches auf der aktuell ausgewählten Seite die Methode *tick()* aufruft. Der Grund, weshalb es nur ein Intervall in der Wurzelkomponente gibt und dass nicht in jeder Komponente ein Intervall gestartet wird, wo es grade gebraucht wird, ist, dass Intervalle asynchron sind und asynchroner Code generell fehleranfälliger als synchroner Code ist. Durch diese Zentralisierung wird die Gefahr ein wenig reduziert. Außerdem müssen so nicht explizit Intervalle schlafen gelegt werden, wenn der Nutzer in einen anderen Menüpunkt wechseln möchte.

Spielfeld

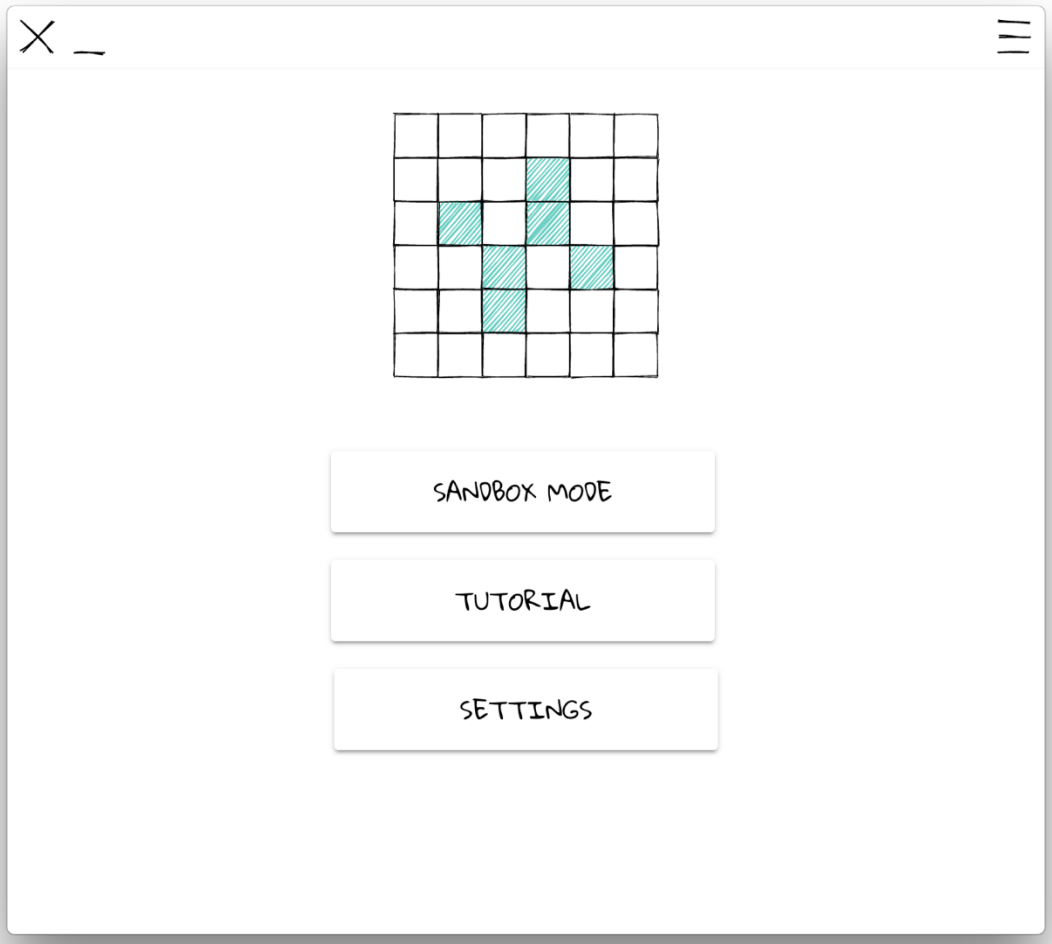
Das Spielfeld zu Conway's Game of Life ist selbst eine Web Komponente und kapselt die vollständige Spiellogik in sich selbst. Komponenten, die das Spielfeld verwenden können durch *calculateNextGeneration()* das Spielfeld die nächste Generation der Zellen berechnen und darstellen lassen. Die Implementierung der Spiellogik wurde mit *ndarray* durchgeführt, womit ein zweidimensionales Array erzeugt wurde, welches die Zellzustände repräsentieren sollte. Hierbei wurden die Nachbarn der Zellen in der Iteration t abgezählt und auf dessen Basis die Zellzustände für die Iteration t+1 gebildet. Die Sterbe- und Belebungsregeln sind frei konfigurierbar durch die Übergabe eines Arrays mit 9 Einträgen, wobei jeder Index die Zahl der lebenden Nachbarn repräsentiert. Ebenfalls kann die Farbe für lebende Zellen, wie auch die Rauheit der Zeichnungen konfiguriert werden.

Gestaltung der Benutzeroberfläche

Für die Benutzeroberfläche habe ich mich entschieden einen selbstgezeichneten Stil zu verwenden. Bei rough.js handelt es sich um eine JavaScript-Bibliothek, die genau das ermöglicht. Die meisten in der Benutzeroberfläche zu sehenden visuellen Komponenten verwenden diese Bibliothek intern.

Hauptmenü

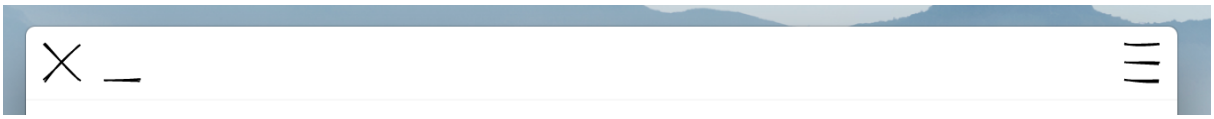
Nachdem die Anwendung gestartet wird, erscheint als erstes das Hauptmenü der Anwendung. Von hier aus können durch das Drücken auf die jeweiligen Schaltflächen die verschiedenen Modi erreicht werden.



Hierbei wurde oberhalb der Schaltflächen ein nicht editierbares Spielfeld platziert. Das Spielfeld wird mit der Figur „clock“ befüllt und durch die *tick()*-Methode, welche vom Hauptintervall aufgerufen wird animiert.

Window Frame

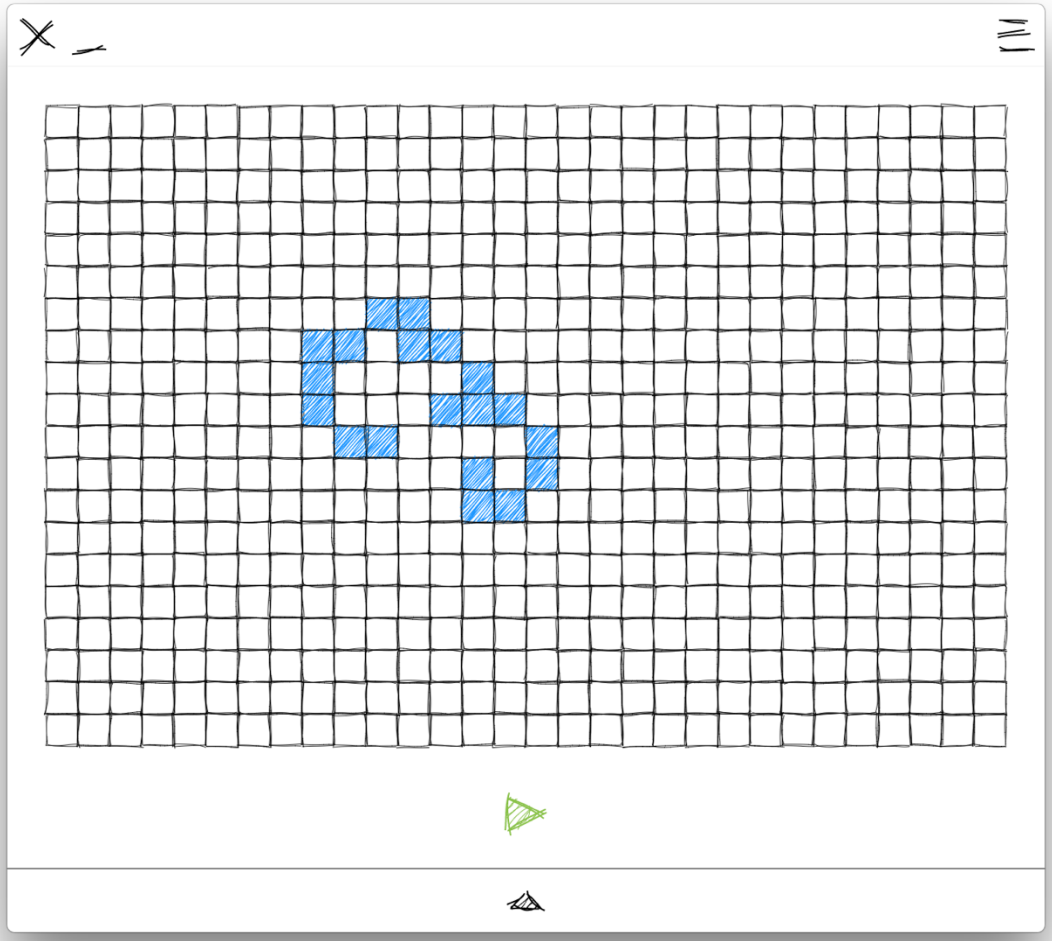
Electron ermöglicht es Fenster zu erzeugen, die nicht den vom Betriebssystem erdachten Fensterrahmen verwenden. Wird der Fensterrahmen entfernt, so sollte Programmierer selbst einen mit HTML, CSS und JavaScript erstellten Fensterrahmen zur Verfügung stellen, welcher dann durch click-events Aufrufe an Electron stellt, welche dann an das Betriebssystem weitergeleitet werden damit das Fenster beispielsweise minimiert oder geschlossen werden kann.



Auf der linken Seite des Fensterrahmens habe ich ein Icon platziert, welches das Fenster beim draufklicken schließt und eins, welches das Fenster minimiert. Auf der rechten Seite befindet sich der Menü-Button. Die Icons wurden mit rough.js erstellt und lassen sich durch die übergebene Rauheit konfigurieren, welche auf der Einstellungsseite angepasst werden kann. Hierdurch wird es dem Nutzer ermöglicht innerhalb des Programms eine visuelle Anpassung des Fensterrahmens durchzuführen.

Sandbox Modus

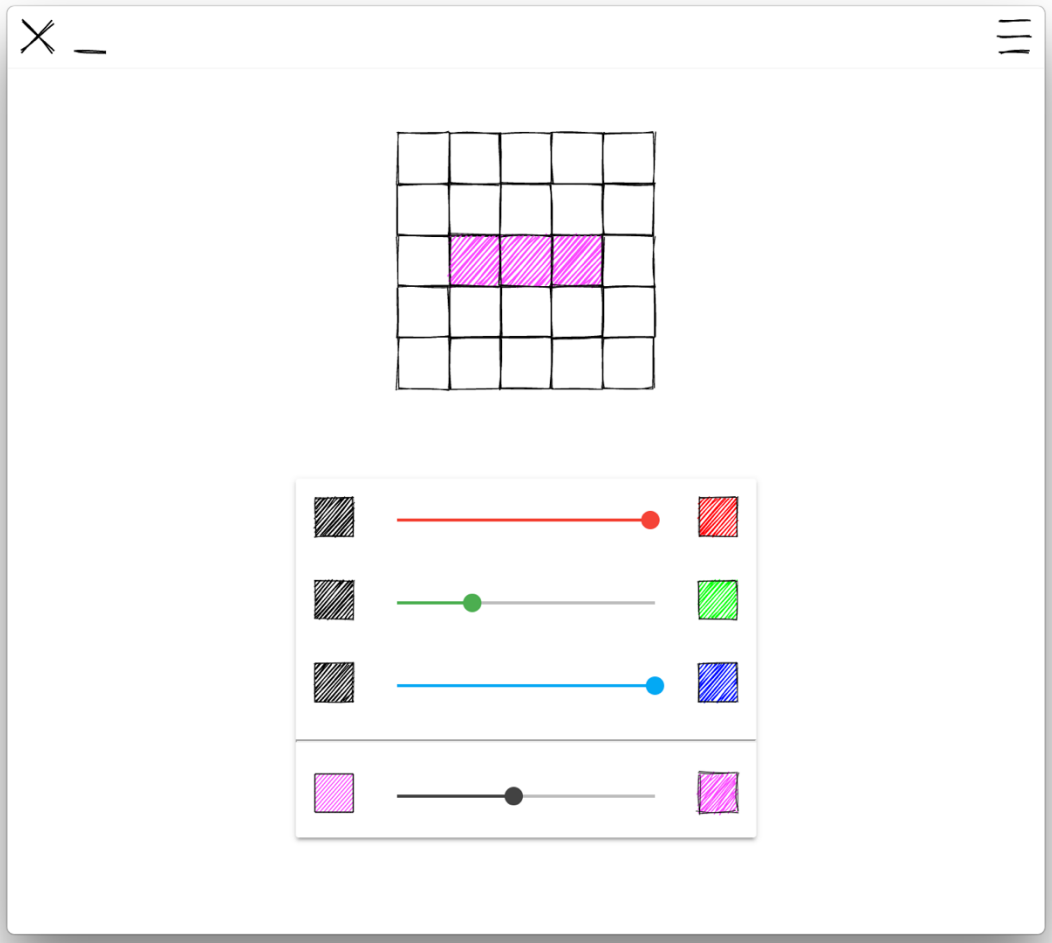
Im steht dem Nutzer ein größeres Feld zur Verfügung. Eine tote Zelle wird lebendig, indem der Nutzer auf sie klickt. Genauso wird beim erneuten Klicken eine lebendige Zelle wieder tot. Ist der Nutzer mit dem Spielfeld zufrieden, kann er den Play-Button drücken. Dadurch wird im Sekundentakt eine neue Generation berechnet und anschließend dargestellt.



Sollte der Nutzer die Regeln für das Sterben und Beleben von Zellen ändern wollen, kann er durch einen Klick auf das Dreieck am unteren Rand des Fensters den Regeleditor ausfahren lassen. Hier lassen sich die Regeln durch das Klicken auf die Kreise anpassen. Der Regeleditor ist von Links nach rechts nach der Anzahl der Nachbarn sortiert. Ein grüner Kreis bedeutet, dass bei der jeweiligen Nachbarzahl eine tote Zelle in der nächsten Iteration wiederbelebt wird und ein roter Kreis bedeutet, dass eine lebende Zelle bei der jeweiligen Nachbarzahl in der nächsten Iteration sterben wird.

Einstellungen

Auf der Einstellungsseite lässt sich die Farbe der lebenden Zellen sowie die Rauheit der Zeichnungen konfigurieren. Unten auf der Seite können die Ausprägungen der Rot-, Grün- und Blaukanäle für die Zellfarbe sowie die Rauheit der Zeichnungen durch das Verschieben von Slidern festgelegt werden. Oberhalb befindet sich ein Spielfeld, welches die Figur „blinker“ darstellt und vom Hauptintervall durch die *tick()*-Methode animiert wird. Das Spielfeld reagiert immer auf die aktuelle Einstellung, die der Nutzer vorgenommen hat. Somit hat der Nutzer ein direktes Feedback, wie sich die Änderungen der Einstellungen später im Sandbox-Modus bemerkbar machen werden.



Links und rechts neben den Slidern befinden sich (nicht editierbare) Spielfelder der Größe 1x1, welche darstellen, wie sich die Positionierung eines Sliders auf die Spielzellen auswirken würde, wenn er nach ganz links bzw. ganz recht verschoben werden sollte. Die Farbkanäle habe ich hierbei allerdings unabhängig voneinander gemacht, da ich die Trennung der einzelnen Kanäle als intuitiver für den Benutzer der Software empfinde, insbesondere wenn dieser ebenfalls ein Programmierer ist.

Tutorial

Zu guter Letzt gibt es noch einen Menüpunkt für die Anleitung, damit neue Nutzer in Conway's Game of Life und die Applikation Cells hineingeführt werden können. Zur bildhaften Verdeutlichung werden hierbei nicht editierbare Spielfelder verwendet.