Technische Dokumentation

Conways Game of Life – byWulf

Autor: Michael Wolf

Inhaltsverzeichnis

1 Einleitung

Durch einen Arbeitskollegen bin ich auf die Code Competitions von IT-Talents aufmerksam geworden. Im Monat Januar 2016 war hier als Aufgabenstellung die Programmierung von „Conways Game of Life" gegeben. Diese klang sehr interessant, sodass ich mich beschloss, für eine Teilnahme am Contest eben diesen Simulator möglichst benutzerfreundlich und zugleich effizient umzusetzen.
Dabei habe ich ein besonderes Augenmerk auf die logische Strukturierung und Aufteilung der einzelnen Bestandteile sowie auf eine effiziente Ausführung geachtet. Weiterhin war mir wichtig, dass das Projekt bei möglichst vielen Usern ohne zusätzliche Installation lauffähig ist.

2 Projektauftrag

2.1 Anforderungen

Von Seiten IT-Talents wurden folgende Anforderungen an das Projekt gestellt:

- Spielfeld und Zellen
  Das Spielfeld ist ein Gitternetz aus X mal X Zellen. Diese Zellen können jeweils zwei Zustände haben: tot, oder lebendig (0 bzw. 1)
  Jede Zelle auf dem Spielfeld hat 8 Nachbarzellen. Wird das Spielfeld am Rand verlassen, so wird es an der gegenüberliegenden Seite wieder betreten. Eine Zelle am Rand des Spielfeldes hat ihre Nachbarzellen also auch auf der gegenüberliegenden Seite.
  Das grundlegende Objekt ist eine Zelle. Diese Zelle muss ihre Position kennen (die Position einer Zelle wird im Verlauf nicht gewechselt, nur der Status) und einen Status haben.
  Tipp: Damit eine "Lebensform" aus dem Spielfeld heraus laufen und am gegenüber liegenden Ende wieder auftauchen kann, muss eine Zelle auch die Grenzen des Spielfeldes kennen.
  Das Spielfeld ist demnach eine Anordnung von allen lebenden und toten Zellen.
- Zyklus
  Das "Game of Life" ist zyklisch angelegt. Das bedeutet, es wird nach und nach immer wieder eine Spielrunde ausgeführt. In jeder Spielrunde werden alle Regeln abgefragt und entsprechend dieser Regeln der neue Status der Zelle (tot oder lebendig) bestimmt.
- Spielregeln
  Die nächste Generation an lebenden Zellen (also der Status aller Zellen in der nächsten Spielrunde) wird durch ein einfaches Regelset berechnet:

1) Jede lebende Zelle, die weniger als 2 lebende Nachbarzellen hat, stirbt
2) Jede lebende Zelle, die 2 oder 3 lebendige Nachbarzellen hat, lebt weiter
3) Jede lebende Zelle, die mehr als 3 lebende Nachbarzellen hat, stirbt
4) Jede tote Zelle, die genau 3 lebende Nachbarzellen hat, wird lebendig

- Ausgabe
  Das Game of Life muss eine grafische Ausgabe haben und es müssen Parameter übergeben werden können. Ob Du dabei eine Ausgabe des Spielfeldes und die Übergabe von Mustern (Erstbesetzung des Spielfeldes) auf der Kommandozeile realisierst, oder ob Du eine grafische Oberfläche erstellst, bleibt Dir überlassen.

2.2 Bewertung

Es wurden folgende zu bewertende Punkte genannt:

- Funktionalität: Lässt sich das Programm ausführen? Tut es, was es soll?
- Code-Qualität: Ist der Code sinnvoll strukturiert und effizient?
- Code-Lesbarkeit: Lässt sich der Quellcode nachvollziehen?
- Dokumentation: Verstehen wir die Bedienung des Programms? Ist der Code kommentiert?
- Zusätzliche Features: Auch besonders aufwändige GUIs oder sinnvolle Zusatzfeatures fließen etwas mit in die Wertung ein ;)

2.3 Zusatzfeatures

Zusätzlich zu der geforderten Grundfunktionalität wurden noch folgende Features implementiert:

- Vorgefertigte Startgenerationen
  Um dem Benutzer einfacher die Verhaltensweisen des Automatens zu präsentieren, wurden diverse Startgenerationen zur Auswahl durch den Benutzer implementiert. Er muss sie lediglich aus einer Selectbox auswählen und schon kann er mit der Ausführung beginnen.
  Weiterhin hat der Benutzer aber natürlich trotzdem die Möglichkeit, eine eigene Startgeneration durch Klicken auf die Zellen festzulegen.
- Unterschiedliches Regelwerk
  Wie auf der Wikipediaseite zu „Conways Game of Life" nachzulesen ist, gibt es die Simulation auch mit abgewandelten Regeln. Diese können ebenfalls in der auf Wikipedia verwendeten Schreibweise „23/3" eingetragen werden und werden anschließend bei der Ausführung beachtet. Dem Benutzer werden dabei schon beliebte und besondere Regelwerke genannt, die er verwenden kann.
- Steuerung der automatischen Generationsentwicklung
  Der Benutzer kann entweder selbst Schritt für Schritt durch die Generationen springen, oder mit einstellbarer Geschwindigkeit die Generationen selbstständig

entwickeln lassen.
- Umfangreiche Hilfe
  Um dem Benutzer die Bedienung des Programms so einfach wie möglich zu gestalten, hat er Zugriff auf eine im Programm eingebaute Hilfe, in der sowohl das Prinzip hinter Conways Game of Life, als auch die Bedienung des Programms erläutert wird.

3 Projektdurchführung

3.1 Technologie

3.1.1 Programmiersprachen

Zur Entwicklung meiner Implementation von „Conways Game of Life" wurde die Programmiersprache „JavaScript" (plus natürlich „HTML" und „CSS") verwendet. Dies hat den Vorteil, dass das Programm auf jedem Rechner mit grafischer Benutzeroberfläche und installiertem Browser sowie auch auf den meisten Smartphones lauffähig ist.
Gerade bei der rechenintensiven Entwicklung der Generationen ist hierbei die unterschiedliche Implementierung der JavaScript-Engines in den verschiedenen Browsern bemerkbar geworden. Chrome hat hier als einziger Browser einen deutlichen Vorsprung und kann am schnellsten die Generationsübergänge berechnen und darstellen.

3.1.2 Entwicklungsumgebung

Als Entwicklungsumgebung wurde „PhpStorm" verwendet. Dies ist momentan die führende IDE für Webentwicklung. Auch wenn in diesem Projekt kein PHP verwendet wurde, kann diese Entwicklungsumgebung wunderbar für reine JavaScript-Projekte verwendet werden. Durch die integrierte Autovervollständigung und Refaktorierungsmethoden ist eine zügige Entwicklung einfach durchzuführen.

3.2 Dateistruktur

Um die Dateien im Projekt logisch zu trennen und so das Projekt wartbar zu halten, wurden die Dateien in folgende Ordner eingeteilt:

- externalLibs/ Alle verwendeten externen Frameworks und Plugins
- css/ Stylesheet-Dateien für die Formatierung
- img/ Bilder zur Anzeige im Programm
- src/ Die Logik des Programms in Form von JavaScript-Code
- startForms/ Die unter „Zusatzfeatures" angesprochenen vorgefertigten Startgenerationen
- index.html Startdatei des Projekts und zugleich HTML-Gerüst des

Benutzerinterfaces

3.3 Klassen

Zur Implementierung der Conway-Automaten wurden die nachfolgenden Klassen entwickelt. Die Beziehung dieser Klassen lässt sich wie folgt zusammenfassen: Die „World" besitzt ein „Ruleset" (=Regelwerk, wann eine Zelle sterben muss und wann wiederbelebt wird) und sehr viele „Cells" (=Einzelne Zelle, die den Zustand „tot" und „lebendig" kennt und die Darstellung regelt). Die „World" übernimmt dabei die Berechnung der neuen Zustände der Zelle nach einem Generationswechsel, da sie alle Zellen kennt.

3.3.1 World

Die Welt speichert Regeln, Startgeneration und die einzelnen Zellen und sorgt dafür, dass sich die Zellen korrekt entwickeln.

Diese Klasse beinhaltet folgende wichtige Attribute:

- cells
  Alle Zellen-Objekte als mehrdimensionales Array, um einfacher mit x/y-Koordinaten auf die jeweilige Zelle zugreifen zu können
- ruleset
  Das zu verwendende Ruleset
- startForm
  Die anzuzeigende Startgeneration, welche bei der Generierung der Zellenobjekte schon direkt angewendet wird

Diese Klasse beinhaltet folgende wichtige Methoden:

- createCells()
  Erzeugt alle Cell-Objekte und legt sie im Array „cells" in zweidimensionaler Form (cells[y][x] = new Cell()) ab. Die Methode gibt den Zellen auch direkt ihren Startzustand anhand der hinterlegten StartForm. Sie sammelt zunächst den reinen HTML-Code jeder Zelle und fügt ihn anschließend als Gesamtpaket in den HTML-DOM ein. Dies ist deutlich effizienter als jede Zelle einzeln einzufügen.
- this.getCell(x, y)
  Gibt die Zelle an der jeweiligen x/y-Position zurück. Das besondere hierbei ist, dass auch Koordinaten angegeben werden können, welche über den Rand hinausgehen. In diesem Fall gibt die Methode automatisch die Zelle zurück, die man bekommt, wenn man am anderen Ende wieder in die Welt eintritt.
- getLivingSurroundings(x, y)
  Ermittelt die lebenden Nachbarn einer Zelle an gegebener x/y-Position.

- this.generate()
  Die wichtigste Methode im ganzen Projekt. Sie ist in zwei Schritte aufgeteilt:
  1. Schritt: Ermitteln, welche Zellen in der nächsten Generation sterben müssen und welche wiederbelebt werden. Hierbei werden alle Zellen durchgegangen und mithilfe der getLivingSurroundings-Methode und dem gespeicherten Ruleset geprüft, ob die Zelle sterben muss oder wiederbelebt wird. Dieser Zwischenzustand „dying" bzw. „resurrecting" wird dann in die jeweilige Zelle gespeichert, ohne ihren echten Zustand zu verändern.
  2. Schritt: Anwenden der ermittelten Zwischenzustände: Erst nachdem von allen Zellen der Übergangszustand ermittelt werden konnte, kann nun der echte Status jeder Zelle aktualisiert werden. Hierzu wird wieder jede Zelle durchgegangen und anhand des Zwischenzustands die Zelle getötet oder wiederbelebt. Die Zelle kümmert sich anschließend um die korrekte Darstellen des neuen Zustandes im Browsern

3.3.2Cell

Die Zelle stellt eine einzelne Zelle in der Welt dar und speichert ihren Zustand „tot" oder „lebend" sowie einen Zwischenzustand „sterbend", „auferstehend" oder „gleichbleibend". Außerdem sorgt sie für die korrekte Darstellung ihres Status auf der Webseite.

Diese Klasse beinhaltet folgende wichtige Attribute:

- alive
  Zustand der Zelle, ob sie lebt (true) oder tot ist (false)
- resurrecting
  Zwischenzustand einer Zelle, wenn sie tot ist und in der nächsten Generation wiederbelebt wird
- dying
  Zwischenzustand einer Zelle, wenn sie lebt und in der nächsten generation sterben wird

Diese Klasse beinhaltet folgende wichtige Methoden:

- setAlive(newAlive)
  Setzt den Zustand der Zelle. Bei true lebt die Zelle, bei false ist die Zelle tot. Diese Methode sorgt direkt für die korrekte Darstellung ihres neuen Zustandes auf der Webseite.

3.3.3Ruleset

Diese Klasse beinhaltet folgende wichtige Attribute:

- surviveArray
  Array, welches als Elemente die Anzahl an lebenden Nachbarn enthält, bei denen eine lebende Zelle in der nächsten Generation weiterleben wird
- resurrectArray
  Array, welches als Elemente die Anzahl an lebenden Nachbarn enthält, bei denen eine tote Zelle in der nächsten Generation wiederbelebt wird

Diese Klasse beinhaltet folgende wichtige Methoden:

- init
  Beim Erstellen eines Rulesets muss ein String in der Form „23/3" übergeben werden. Diese Methode parsed nun diesen String, prüft ihn auf Gültigkeit und wandelt ihn in die zwei o.g. Arrays um. Sollte es dabei zu einem Fehler kommen (String hat falsches Format oder ungültiges Zeichen oder zu hohe Zahl (>8) erkannt, wird ein Fehler geworfen)
- this.willDie(livingSurroundings)
  Bei gegebener Anzahl an lebenden Nachbarn gibt diese Methode zurück, ob eine lebende Zelle unter diesen Bedingungen sterben muss
- this.willResurrect(livingSurroundings)
  Bei gegebener Anzahl an lebenden Nachbarn gibt diese Methode zurück, ob eine tote Zelle unter diesen Bedingungen wiederbelebt wird

3.3.4StartForm

Eine StartForm gibt an, welche Zellen zu Beginn leben sollen und welche tot sein sollen. Hierbei können beim Erzeugen einer neuen Startform zwei verschiedene Werte zum Festlegen der Startgeneration übergeben werden:

- Typ Funktion
  Callbackfunktion, welche mit x/y/width/height als Parameter für jeder Zelle aufgerufen wird und true (lebend) oder false (tot) zurückgeben muss. Damit kann bspw. die RandomStartForm oder ein in der Mitte der Welt zentriertes Objekt definiert werden.
- Typ String
  Pattern-String, wo jede Zeile für die y-Position und jedes Zeichen pro Zeile für die x-Position steht. Jedes Zeichen darf dabei den Wert „ „ (Leerzeichen) oder „X" haben. Ein „X" steht dabei für „lebend", jedes andere Zeichen (oder nicht definiert) steht für „tot".

Diese Klasse beinhaltet folgende wichtige Methoden:

- this.isAlive(x, y, width, height)
  Diese Methode wird beim Generieren der Zellen in der Klasse Worlds (Methode createCells()) für jede einzelne Zelle aufgerufen und gibt den durch die im Constructor übergebene Funktion oder dem Pattern-String ermittelten Lebend-Zustand zurück

3.3.5App

Diese Klasse stellt das Programm an sich dar, das der Benutzer bedienen kann. Sie initialisiert das Frontend, sorgt dafür, dass Klicks im Frontend bestimmte Ereignisse auslösen und stellt quasi die Verbindung der Welt mit dem Benutzer her. Es erzeugt die World und gibt dem Benutzer die Möglichkeit, die Startgeneration und das Regelwerk auszuwählen.

Die Klasse gibt dem Benutzer außerdem die Möglichkeit, das automatische Erzeugen neuer Generationen zu starten/pausieren und dabei die Geschwindigkeit festzulegen. Dabei wird intern in festen Abständen die Methode „generate()" der Klasse „World" aufgerufen.

3.4 Abgrenzung zu Fremdleistungen

In diesem Projekt wurden zur einfacheren und effizienteren Entwicklung folgende Frameworks und Plugins verwendet:

- jQuery
  Dieses JavaScript-Framework ist im Standardrepertoire eines jeden JavaScript-Projektes vorhanden (mit wenigen Ausnahmen) und dient als Grundstein in der heutigen JavaScript-Entwicklung. Es ermöglicht dem Benutzer einen sehr einfachen und crossbrowserfreundlichen Zugriff und Manipulation von HTML-Elementen und stellt noch weitere Bequemheiten zur Verfügung, welche im reinen JavaScript nur umständlich umgesetzt werden könnten.
- Bootstrap
  Auch dieses Frontend-Framework wird in immer mehr Projekten verwendet. Es bringt ein umfangreiches und ebenfalls crossbrowserfreundliches Grundgerüst an HTML-Strukturen mit und sorgt dafür, dass man sehr einfach und schnell umfangreiche und wohl designte HTML-Frontends hochziehen kann.
- bootstrap-slider
  Zum Einstellen der Geschwindigkeit der automatischen Generationsentwicklung wird dem Benutzer ein Schieberegler angeboten, welcher auf dieses Plugin zurückgreift. Ein Schieberegler ist für diesen Use Case für den Benutzer eine bequeme Möglichkeit zur Eingabe, welche normales HTML leider nicht zur Verfügung stellt. Hier hätte man auf ein numerisches Eingabefeld zurückgreifen müssen, welches bei

weitem nicht so viel Komfort bieten würde.

4 Lessions learned

Durch die Umsetzung dieses Projekts konnte ich wichtige Dinge lernen und mein fundiertes Wissen in Entwicklung weiter ausbauen.

So habe ich bei meiner Recherche einiges über Conway und seine Arbeit als Mathematiker herausgefunden und konnte außerdem viel neues über die Funktionsweise von Automaten lernen.

Ich habe feststellen müssen, wie unterschiedlich die verschiedenen Browser bzw. deren JavaScript-Implementierung in der Ausführungsgeschwindigkeit sind und dass Chrome in diesem speziellen Anwendungsfall die Nase deutlich vorne hat.

Zugleich konnte ich das ewig vernachlässigte Thema Dokumentation sowohl im Quellcode als auch in Form dieser technischen Dokumentation üben und anwenden und freue mich über Feedback, was nicht so gut war und was ich noch besser machen kann.

Nur durch Feedback kann man sich als Programmierer weiterentwickeln und seine Fähigkeiten verbessern. Aus diesem Grund nehme ich auch an diesem Wettbewerb teil, um möglichst viel und erfahrenes Feedback zu erhalten.

5 Quellenverzeichnis

- https://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens
- https://www.it-talents.de/cms/aktionen/code-competition/code-competitions
- https://jquery.com/
- http://getbootstrap.com/
- https://github.com/seiyria/bootstrap-slider