Dokumentation

Software-Projekt: Conways Game Of Life

Software Entwicklung 2 (WS 20/21)

Dominik Metzler	MatrNr.: 40295	MI
Johannes Niedworok	MatrNr.: 40183	MI

Stand: Stuttgart, den 17.02.2021

Inhaltsverzeichnis

Inhaltsverzeichnis **2**

1 Einleitung **3**

2 Architektur **5**

2.1 Use Case Diagramm, UML und Sequenzdiagramm 5

2.2 Engine 7

2.3 UI Framework (MVC) 7

3 Features **8**

3.1 Presets 8

3.2 Generations-Steuerung 9

3.3. View-Einstellungen 9

1 Einleitung

Das Spiel “Conway’s Game Of Life” wurde 1970 von dem Mathematiker John Horton Conway (*1937-†2020) entworfen. Dabei handelt es sich um einen zweidimensionalen zelluläre Automaten.

Das Spielfeld besteht aus einem unendlichen *Grid,* wobei jedes Feld (*Cell*) die beiden Zustände “*dead”* oder *“alive”* annehmen kann (s. Bild 1 *alive*=lila, *dead*=grey)

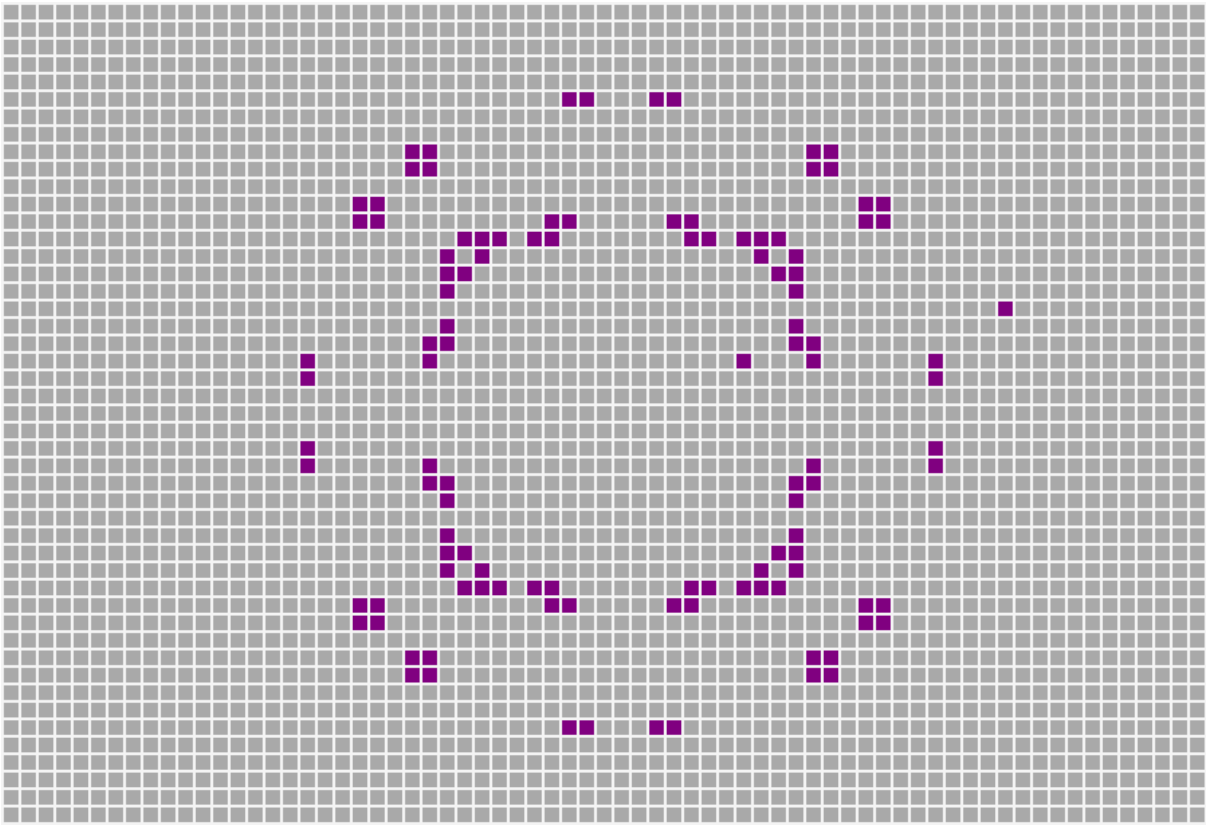


Bild 1- Spielfeld

Jede *Cell* besitzt acht *Neighbor-Cells*. Der Zustand dieser *Cells* wird bei einem Generationswechsel auf Grundlage der folgenden Regeln berechnet:

Regel 1:
“Eine tote Zelle mit genau drei lebenden Nachbarn wird in der Folgegeneration neu geboren.”

Regel 2:
“Lebende Zellen mit weniger als zwei lebenden Nachbarn sterben in der Folgegeneration an Einsamkeit.”

Regel 3:
“Eine lebende Zelle mit zwei oder drei lebenden Nachbarn bleibt in der Folgegeneration am Leben.”

Regel 4:
“Lebende Zellen mit mehr als drei lebenden Nachbarn sterben in der Folgegeneration an Überbevölkerung.”

Das Programm, das im Rahmen dieser Projektarbeit entwickelt wurde, stellt eine visualisierte und automatisierte Simulation dieser Regeln dar, und fügt ihr diverse Bedienelemente, zur Steuerung der Generationen, hinzu, sowie eine Auswahl an Presets.

2 Architektur

Im Folgenden werden die einzelnen Bestandteile des Programms genauer erläutert.
Unsere API ist als Javadoc im Ordner docs/apidocs im Repo vorhanden. UML und Sequenzdiagramm liegen als PDF im Ordner docs.

2.1 Use Case Diagramm, UML und Sequenzdiagramm

Use-Case-Diagramm:

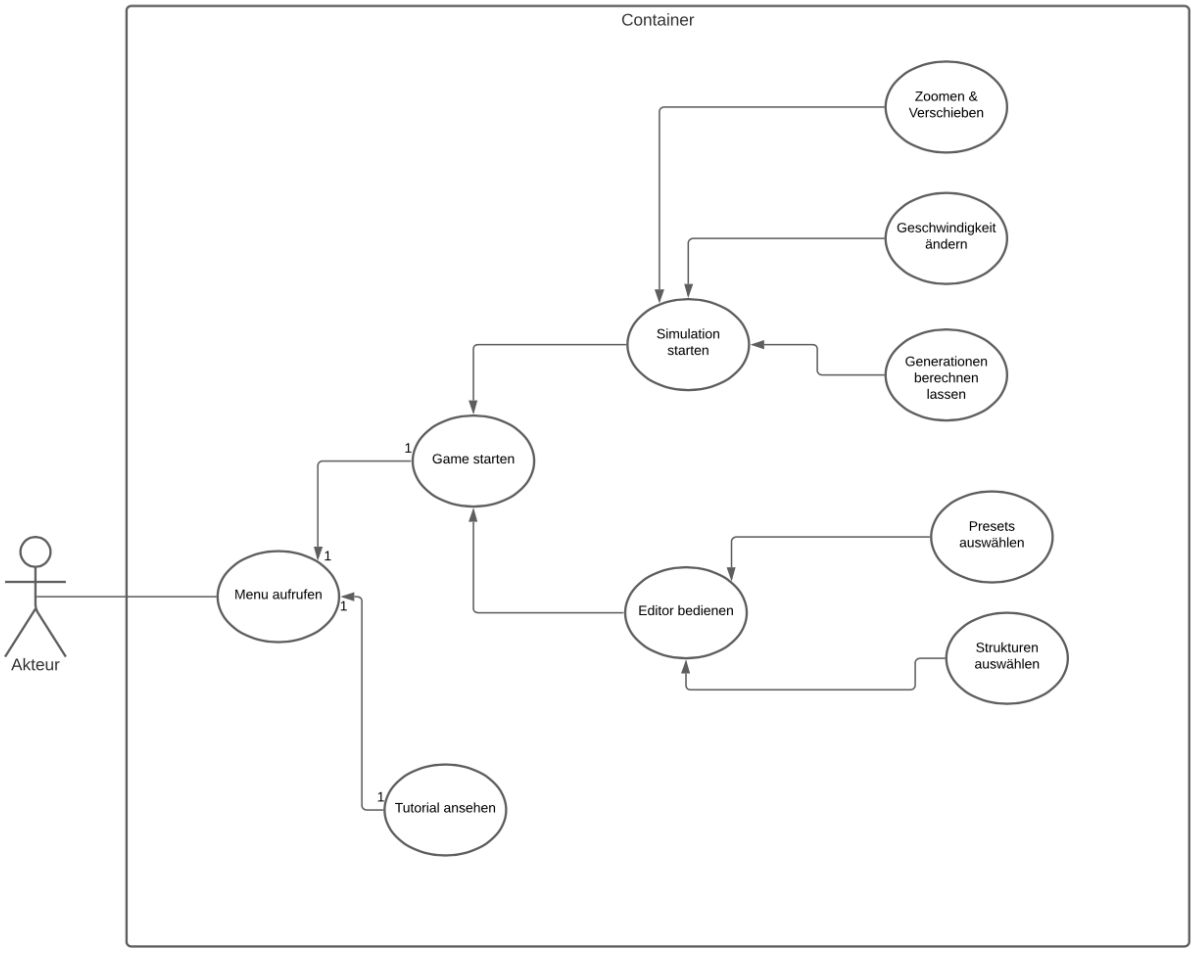


Bild 2 - Use Case Diagramm

UML:

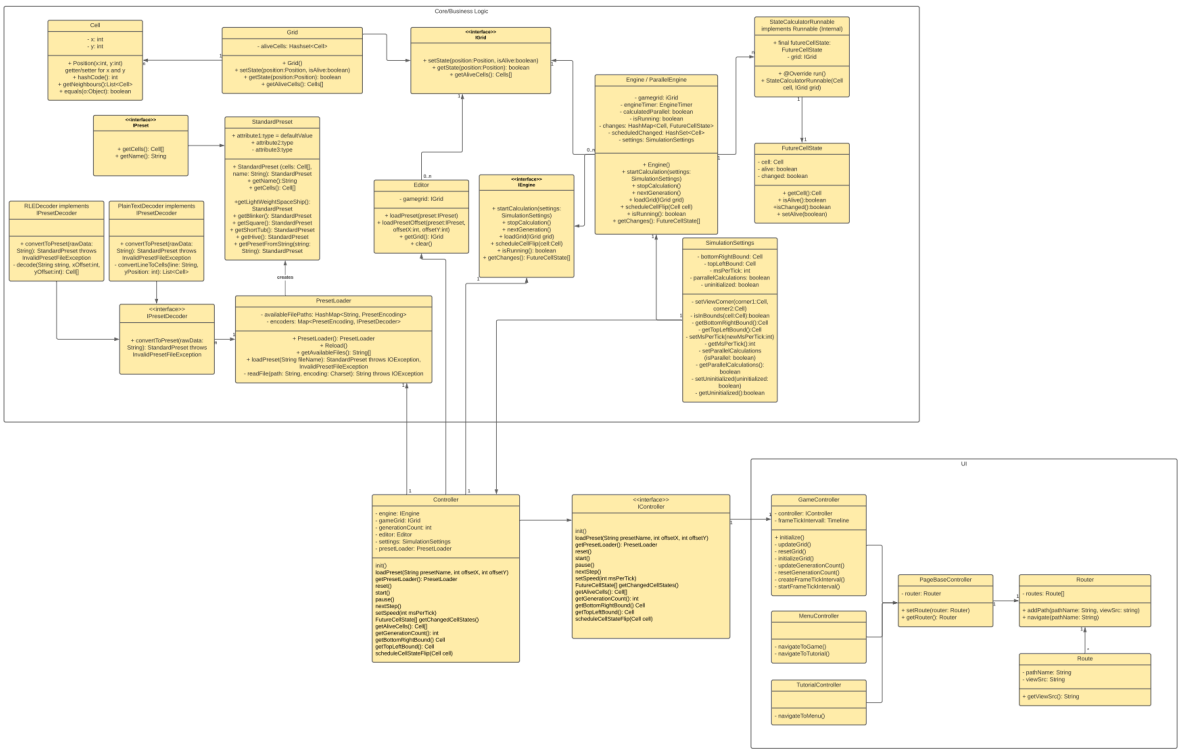


Bild 3 - UML

Sequenzdiagramm:

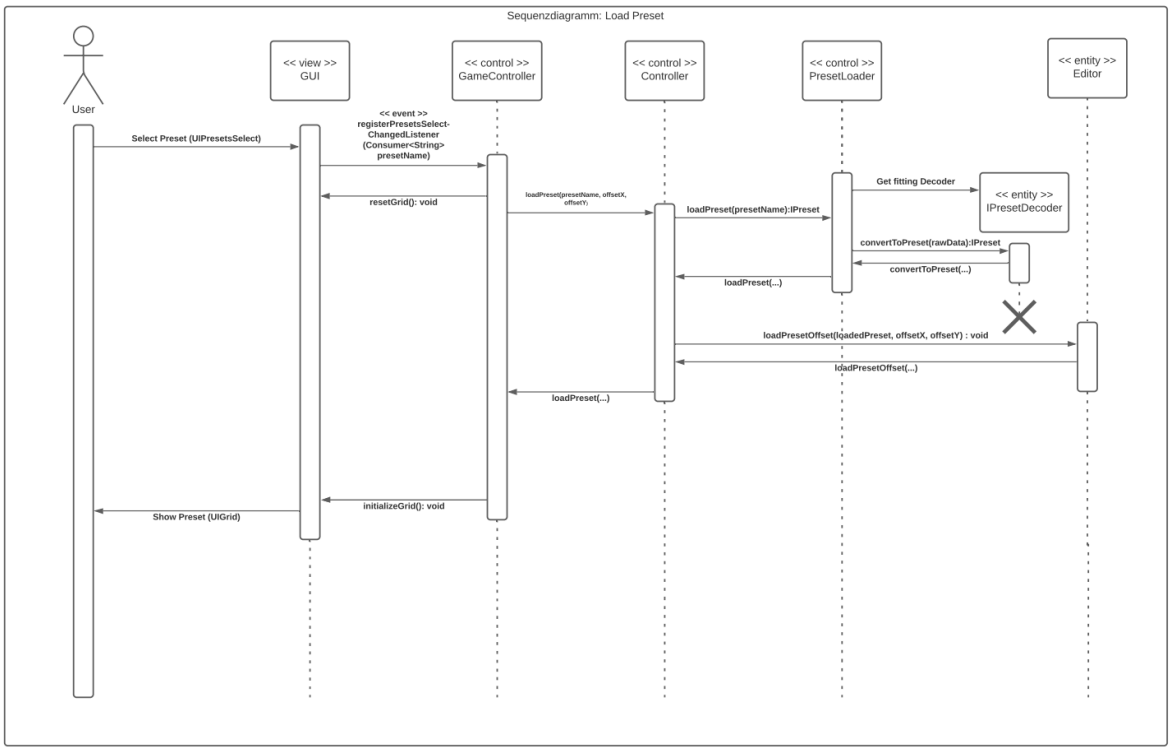


Bild 4 - Sequenzdiagramm

2.2 Engine

Den Kern der Berechnungen stellt die Engine dar. Durch Verwendung des Interfaces IEngine wird Erweiterbarkeit und Austauschbarkeit gewährleistet. Engines können durch die EngineFactory per Reflection (mit ihren absoluten Klassennamen) oder (im Fall der beiden eingebauten Lösungen) per Name geladen werden.
Um die Engine herum existieren einige weitere Elemente zur Berechnung von neuen Generationen und dem modelieren des aktuellen Zustands:
Cell: Stellt eine einzelne Zelle auf dem Grid dar
Grid (IGrid): Die Gesamtheit aller lebenden und toten Zellen
FutureCellState: Eine Art der Engine während den Berechnungen Zukünftige und momentane Zustände zu unterscheiden und Änderungen zu speichern

EngineTimer: Ein Wrapper für `java.util.concurrent.ScheduledExecutorService` mit PoolSize=1

SimulationSettings: Bestimmen Werte wie die Geschwindigkeit, oder ob die Engine parallel oder synchrone Berechnungen verwenden soll.

IPreset: Kann per Editor auf ein IGrid geladen werden.
PresetLoader: Kann Presets aus dem Arbeitsverzeichnis laden. Unterstützt RLE und PlainText (Siehe 3.1).

2.3 UI Framework (MVC)

Model: `de.hdm_stuttgart.mi.gameoflife.core` (s. Engine)
Views (FXML): `resources/views`
Controllers: `de.hdm_stuttgart.mi.gameoflife.controllers`

Ein wichtiger Grundbaustein des UI Frameworks ist das **Router-Package** `de.hdm_stuttgart.mi.gameoflife.controllers.router`. Es dient zur flexiblen Navigation zwischen einzelner Views.
Diese Views sind i.d.R. an einen Controller gebunden:

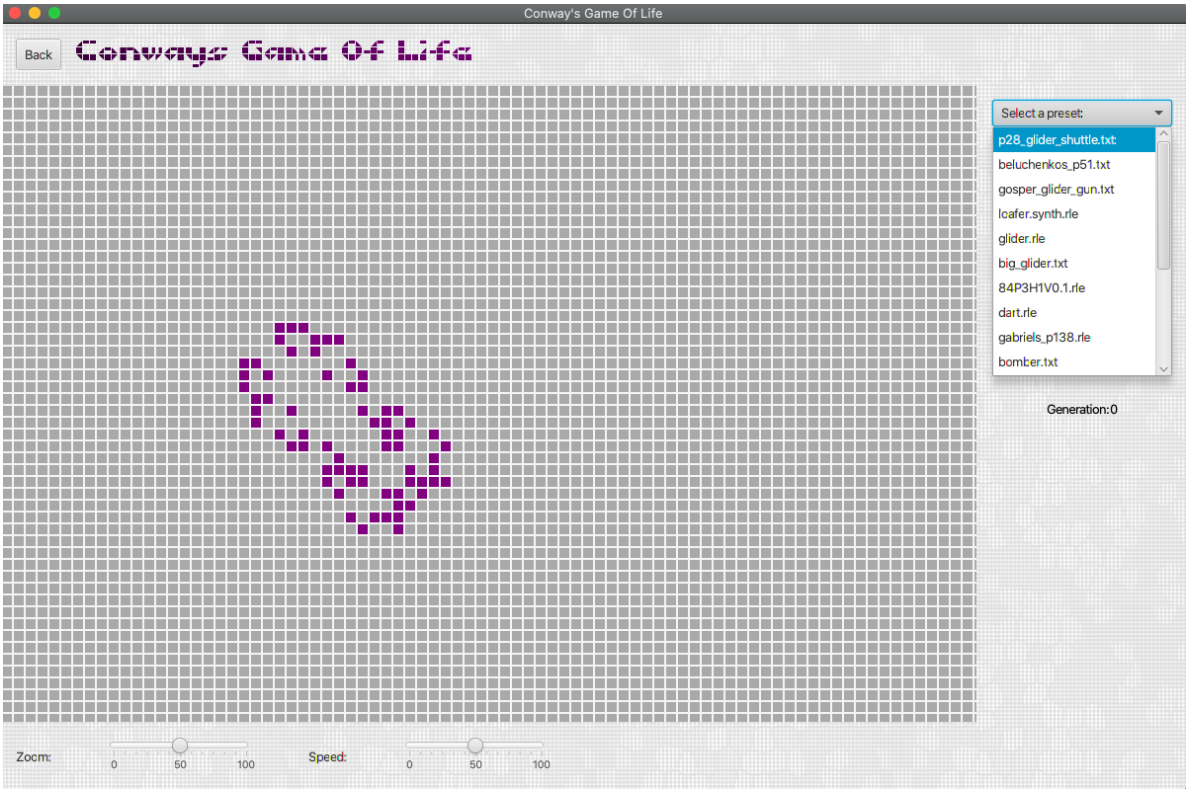
Menu-View & MenuController: Startseite mit Menu
Turorial-View & TurorialController: Auflistung der Spielregeln
Game-View & GameController: Spielfeld mit Bedienelementen (Hauptansicht)

Das Package `de.hdm_stuttgart.mi.gameoflife.controllers.components` stellt eine Reihe von unabhängigen **UI-Components** bereit, die hauptsächlich im GameController verwendet werden (bspw.: UIZoomSlider, UIPresetSelect)

3 Features

3.1 Presets

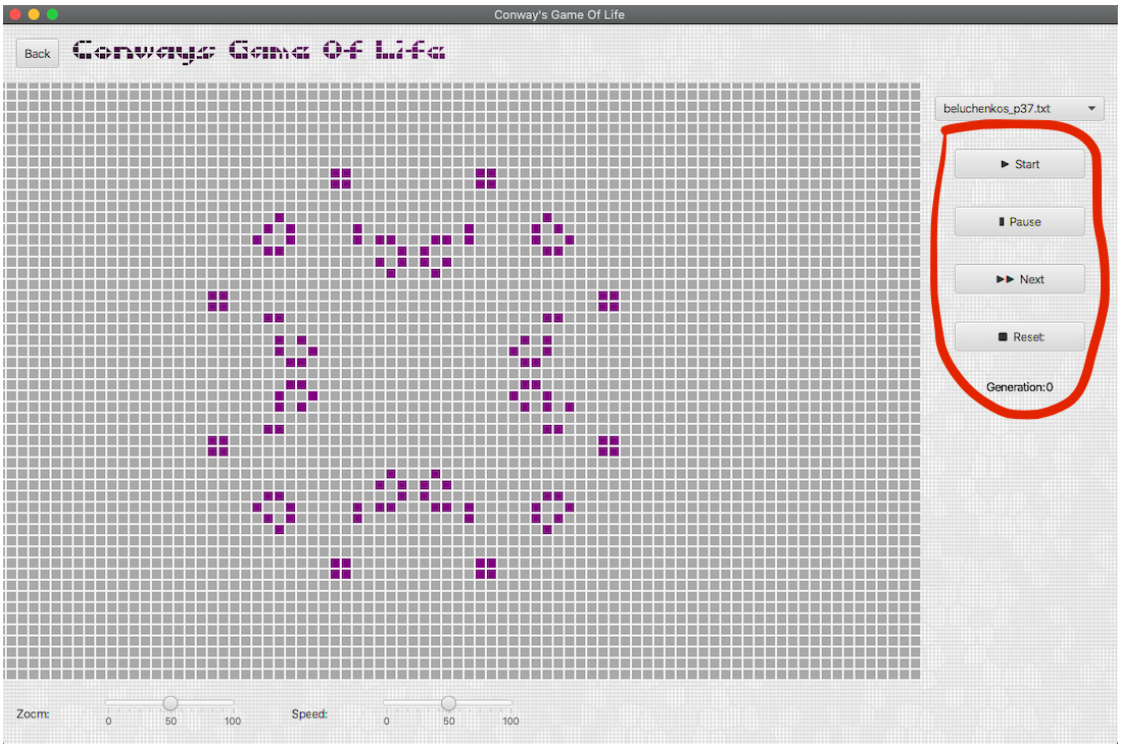
Auswahl 17 einzigartiger Presets



Die Presets werden in den für Game of Life gängigsten offen dokumentierten Formaten Plaintext und RLE codiert, wobei Plaintext eine einfache 1:1-Speicherung von lebenden und toten Zellen als O und . innerhalb von Zeilen, und RLE eine erweiterte Version des allgemeinen Run-Length-Encodings nach dem Format <Run-Length><Symbol: b oder o> darstellt. Beide Formate erlauben Kommentar-Zeilen beginnend mit ! im Fall Plaintext und # im Fall RLE. RLE unterstützt zudem das Verschieben des Patterns mit einer Zeile “#R xOffset yOffset”. Beide Formate lassen eine Benennung des Presets zu:
#N RLE Preset Name
!Name: Plaintext Preset Name
Das Format wird automatisch anhand der Dateiendung (.txt oder .rle) erkannt.

3.2 Generations-Steuerung

Mit Hilfe der markierten Bedienelemente kann die Simulation gestartet, gestoppt oder gelöscht werden.



3.3. View-Einstellungen

Zoom-Slider: Vergrößerung/Verkleinerung des Spielfelds
Speed-Slider: Geschwindigkeit der Generationen