Technology
Arts Sciences
TH Köln

QQ2 Projekt:
Erweitern des GBG Frameworks um einen Delay Slider und eine Logging Funktion

Johannes Kutsch

11090517

Medieninformatik

Prüfer:
Prof. Dr. rer. nat. Wolfgang Konen

16. Mai 2017

Zusammenfassung

Im Rahmen dieses QQ2 Projektes wurde das GBG Framework um zwei Features erweitert. Das erste Feature ist ein Logmanager, der Spiele aufzeichnet und anschließend wiedergeben kann. Außerdem wurde ein Slider implementiert, mit welchem die Simulationsgeschwindigkeit eines Spieles, während der Laufzeit, eingestellt werden kann.

Delay Slider

Über den Delay Slider kann die Simulationsgeschwindigkeit des Frameworkes, während der Laufzeit, schnell verändert werden. Dies vereinfacht das Debuggen bei langen Spielen, da man zuerst mit einer schnellen Simulationsgeschwindigkeit schnell zu fortgeschritteneren Spielzuständen gelangen kann, welche dann bei einer langsamen Simulationsgeschwindigkeit untersucht werden können. Der Minimal- und Maximalwert des Sliders sowie die Startgeschwindigkeit der Simulation kann in der Klasse `Arena.java` eingestellt werden können.

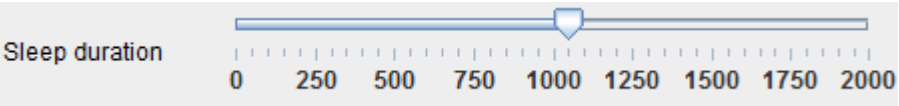

Abbildung 1: Der Delay Slider

Logging und Replay von Spielen

Die Logging- und Replayfunktionen wurden durch einen selbstgeschriebenen Logmanager umgesetzt. Der Logmanager besteht aus den Klassen `LogManager.java` und `LogManagerGUI.java`. Die Klasse `LogManager.java` ist innerhalb des Logmanagers für das Aufzeichnen von Spielen zuständig, während die Klasse `LogManagerGUI.java` ein User Interface bietet über welches die aufgezeichneten Spiele wiedergegeben werden.

Aufzeichnen von Spielen

Um Spiele aufzeichnen zu können, muss eine Instanz der Klasse `LogManager.java` erstellt werden.
Um die Aufzeichnung zu starten muss durch Aufrufen der Funktion `newLoggingSession(StateObservation)` eine neue Session erstellt werden. Als Parameter wird der initiale Spielzustand des Spieles, welches aufgezeichnet werden soll, übergeben. Der Rückgabewerte dieser Funktion ist eine `sessionid` vom Typ int, welche für das Hinzufügen weiterer Spielzustände benötigt wird.
Nachdem eine neue Session erstellt wurde, können nun durch Aufrufen der Funktion `addLogEntry(Types.ACTIONS, StateObservation, int sessionid)` weitere Spielzustände zu dieser Session hinzugefügt werden. Diese Funktion muss dafür bei jedem neuem Spielzustand mit der Aktion, die zu diesem Spielzustand führte, dem neuem Spielzustand und der `sessionid` aufgerufen werden.
Nachdem ein Spiel beendet wurde muss auch die Session beendet werden. Dazu muss die Funktion `endLoggingSession(int sessionid)` mit der `sessionid` aufgerufen werden. Diese Funktion beendet die jeweilige Session (es können also keine weiteren Spielzustände gespeichert werden) und erzeugt eine `.gamelog`-Datei, welche alle für das Replay benötigten Informationen enthält.

Wiedergabe von Aufzeichnungen

Möchte man die Aufzeichnung eines Spieles betrachten, wird die Klasse `LogManagerGUI.java` benötigt. Diese erzeugt ein `JFrame` mit welchem durch eine Aufzeichnung navigiert werden kann. Beim Erzeugen einer Instanz dieser Klasse muss ein `LogManager` sowie ein `GameBoard` übergeben werden.
Der `LogManager` wird dazu benötigt Dateipfade für die `.gamelog`-Dateien zu ermitteln und um Optionen des LogManagers über die Benutzeroberfläche zu ändern. Auf dem `GameBoard` wird die Aufzeichnung des Spieles dargestellt. Hierbei ist es wichtig, dass das `GameBoard` zu dem Spiel gehört, welches man betrachten möchte.

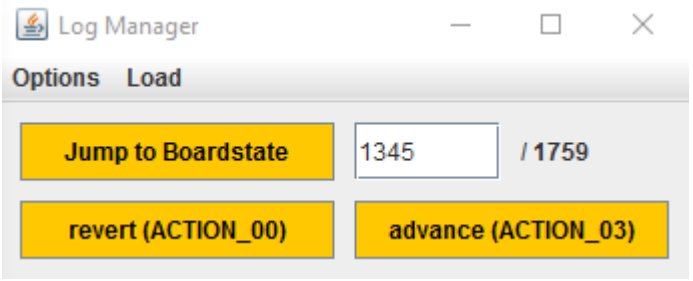

Abbildung 2: GUI des Log Managers

Im `JFrame` kann nun im Menü *Load* → *Load Gamelog* eine `.gamelog`-Datei ausgewählt werden, welche wiedergegeben werden soll. Anschließend kann die Aufzeichnung, mithilfe der Buttons *advance* oder *revert*, durchlaufen werden. Es ist außerdem möglich mit dem Button *Jump to Boardstate* direkt zu einer Position innerhalb der Aufzeichnung zu springen.

Advanced Logging

Beim advanced Logging handelt es sich um eine Loggingmethode, welche entwickelt wurde, da beim normalen Logging die Daten aller Sessions, welche nicht Ordnungsgemäß durch den Aufruf der Funktion `endLoggingSession()` beendet wurden, verloren gehen. Es würde bei einem Programmabsturz also keine Datei entstehen, mithilfe welcher genauer untersucht werden kann, bei welchem Spielzustand der Fehler auftrat.
Dieses Problem wird beim advanced Logging umgangen, indem jeder Session ein temporärer Ordner zugewiesen wird, in welchem jeder neue Logeintrag sofort gespeichert wird. Wird eine Session Ordnungsgemäß durch den Aufruf der Funktion `endLoggingSession()` beendet, wird aus diesen temporären Logdateien wie gewohnt eine `.gamelog`-Datei erstellt. Sollte das Programm allerdings vor dem Aufruf dieser Funktion beendet werden kann aus den temporären Logdateien mit der Funktion `generateLogSessionContainerFromFile(string path)` oder durch auswählen des Menüpunktes *Options* → *Compile temporary Gamelog* im `LogManagerGUI.java` manuell eine `.gamelog`-Datei erstellt werden.