Titel

Daniel Foehn

17. Dezember 2015

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Andwendungszweck des Dokumentes

1.2 Zielpublikum

1.3 Versionierung

1.4 Glossar

2 Übersicht

2.1 Systemkontext

2.2 Ausgangslage

3 Anforderungen

Es gilt ein Netzwerkfähiges Computerspiel mit dem Namen Dots&Boxes zu entwickeln.

3.1 funktionelle Anforderungen

- Gegen Computer spielbar
- Spielstand kann abgespeichert werden
- Spielstand kann vom Speicherstand geladen werden
- Es kann online nach einem NON-AI Gegenspieler gesucht werden
- Es werden die Regeln von Dots&Boxes angewandt

3.2 nicht-funktionelle Anforderungen

- Es soll keine Angabe der IP Adresse nötig sein.
- Der Code soll gut und brauchbar dokumentiert sein
- Das GUI besteht aus
  - Spielfeld
  - Ersichtlich wer am Zug ist
  - Linien können Spieler zugewiesen werden
  - Das Spielfeld kann in Länge und Höhe variieren

4 Zeitplanung

Abgabe ist in der letzten Semesterwoche (**18. Dez 2015**)

5 Testplanung

5.1 Testumgebung

5.2 Testfälle

5.2.1 Unit Tests

5.2.2 Blackbox Tests

6 Grobdesign

6.1 Design of Punkt XY

7 Detaildesgin

7.1 Umsetzung von Punkt XY

7.2 Zustandsdiagramm

7.3 Sequenzdiagramm

7.4 Klassendiagramm

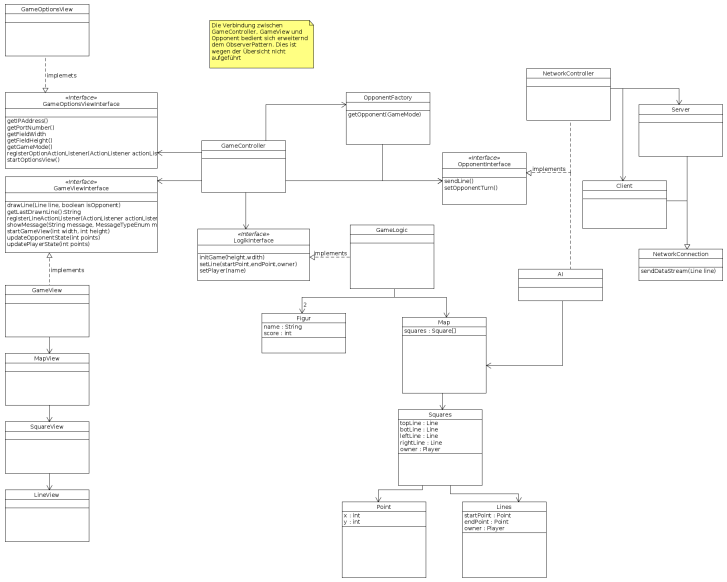

Abbildung 1: class diagram

7.4.1 Class Details

GameController Der GameController ist für den Datenfluss zwischen Game-Logik, View und Netzwerk zuständig.

GameLogic Die GameLogic beinhaltet die Regeln von Dots & Boxes. Sie überprüft die Spielzüge auf ihre Validität und setzt wer als nächster den Zug macht. Sie verwendet zur Datenmodelierung die Klassen *Map, Figur, Squares, Lines* und *Point*

LogikInterface definiert die Schnittstelle zwischen GameLogic und GameController

OpponentFactory Die OpponentFactory verwendet das Factory Pattern und erstellt eine Klasse basierend auf dem OpponentInterface. Sie erstellt ein Objekt der Klasse NetworkController oder AI

OpponentInterface definiert die Schnittstelle zwischen Network/AI und GameController

AI Die AI Klasse beinhaltet die ganze Computerspielerlogik.

GameOptionsView Die GameOptionsView stellt das GUI zur Eingabe der Optionen dar.

GameView Die GameView stellt das Spiel UI dar. Sie verwendet die Klassen *MapView,SquareView* und *LineView* um das Spiel korrekt darzustellen.

GameViewInterface definiert die Schnittstelle zwischen GameController und GameView dar.

GameOptionsViewInterface definiert die Schnittstelle zwischen GameController und GameOptionsView dar.

Appendix