Erläuterungen zu den Klassendiagrammen

Hinweis: *Um dieses Dokument vollständig zu verstehen, ist es notwendig, die zugehörigen Diagramme zu betrachten.*

Die Klassendiagramme folgen dem MVC-Pattern, dabei ist das Modell unabhängig von der View und dem Controller.

Das Klassendiagramm für die View bzw. der Controller des Clients hat die `MdgaApp` Klasse, welche von der `SimpleApplikation` von der jME geerbt und somit der Beginn der Applikation.

Das Interface `AnimationHandler` ist für die Abhandlung der Animationen im Spiel verantwortlich. Dafür gibt es eine abstrakte Klasse `Animation` und verschiedene Ausprägungen, welche die verschiedenen Animationen im Spiel darstellen. Dafür gibt es sechs verschiedene Animationen Klassen: `PlayCard`, `DrawCard`, `Dice`, `MoveAnimation`, `SwapAnimation` und `ThrowAnimation`.

Die Klasse `Dialog` (aus dem Package `jme-common`) ist für die Dialoge zuständig. Dabei existieren in unserem Spiel 5 Dialogs und deren Klassen: `Start`, `Menu`, `Video`, `Sound` und `Network`.

Es existieren zwei Ansichten:
Eine `BoardView`, welche das Spielbrett an sich darstellt, und deshalb auch eine `NodeControl`, `PieceControl` und `CardControl` besitzt, um die Knoten, Figuren und Karten anzupassen. Die Klassen `NodeControl`, `PieceControl` und `CardControl` erben von der Klasse `AbstractAppControl`.
Die `GuiView`, soll die GUI, des Spiels darstellen.

Das Spiel hat AbstractAppStates, um das Spiel in verschiedene Zustände zu unterteilen, d. h. dass diese unabhängig voneinander existieren. Dabei gibt es ein `MdgaState`, `MusicState` und `SoundState`.

Der `ActionSynchronizer` besitzt die Fähigkeit, die Sounds des Spiels zu synchronisieren.

Das Modell wird in drei Teile aufgeteilt.

Das Client-Modell hat zum einen die Client-Messages (siehe Artefakt zu Messages) und den Zustandsautomat für den Client (siehe Artefakte zum Zustandsautomat des Clients). Die `ClientGameLogic` ist für die Logik und die richtigen Zustandsübergänge des Automaten des Clients zuständig.

Das Server-Modell hat zum einen die Server-Messages (siehe Artefakt zu Messages) und den Zustandsautomat für den Server (siehe Artefakte zum Zustandsautomat des Servers).

Die `ServerGameLogic` ist für die Logik und die richtigen Zustandsübergänge des Automaten des Servers zuständig.

Das Game-Modell ist das grundlegende Modell für das Spiel. Die Klasse `Board` soll das Brett an sich modellieren von Spielfeldern. Das Brett ist für die Spielerdaten (Klasse `PlayerData`) und Knoten (als Array modelliert, von der Klasse `Node`) zuständig. Jede Figur besitzt eine Position auf dem Feld und einen Zustand (d. h. wo sich die Figur gerade befindet). Weiter hat eine Spielfigur auch ein Schildstatus (Enum `ShieldState`). Die Klasse `Game` ist für alles Zuständig, d. h. das Management der Spieler, der Statistiken, der Bonuskarten, der TSKs und des Bretts an sich.