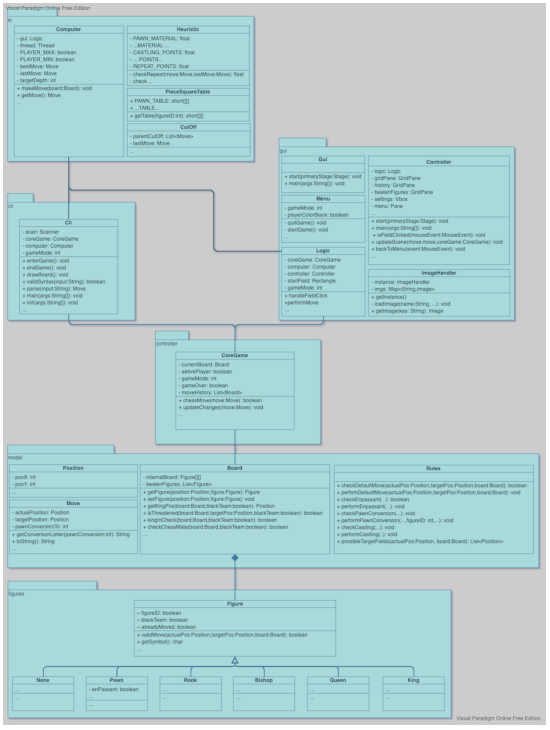
Architektur Dokumentation

Im Folgenden werden wir unsere Architektur aus Nutzersicht sowie aus statischer und dynamischer Schicht darstellen.

Nutzersicht

[siehe Use-Case-Diagramm und Story Cards]

Statische Sicht



[Klassendiagramm statische Sicht][1]

Wie im Klassendiagramm sichtbar, orientieren wir uns an dem Architektur-Pattern der Schichtenarchitektur und dem Design-Pattern MVC (Model-View-Controller).

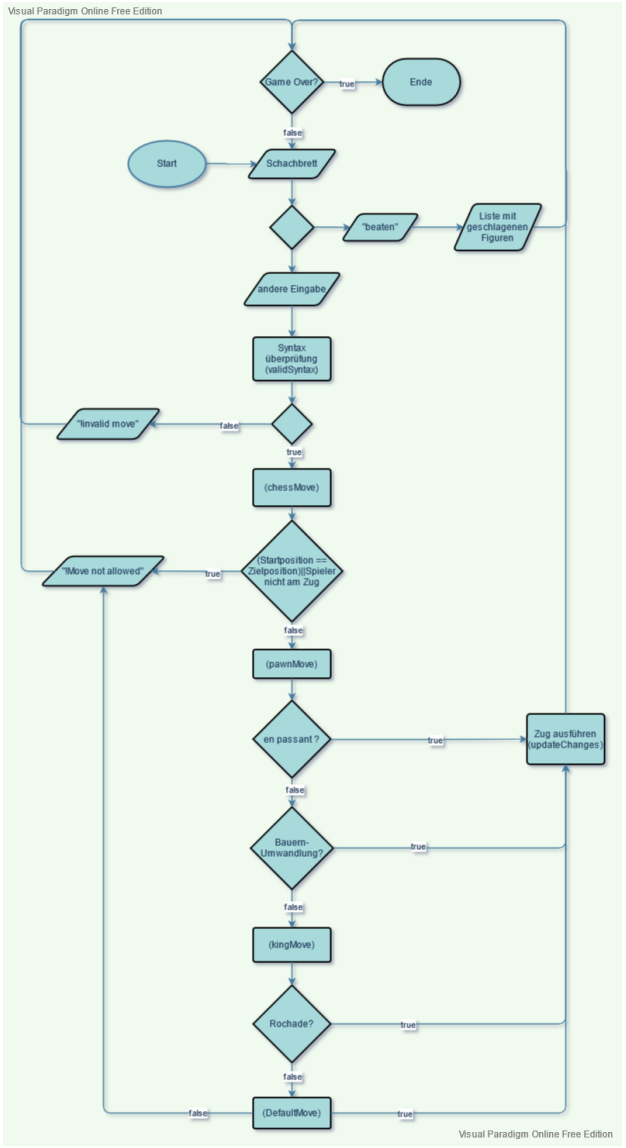
- View: Die Klassen CLI und GUI übernehmen dabei die Aufgabe der Darstellung nach außen auf der Konsole bzw. als grafische Schnittstelle.
- Model: Die Klassen Position, Move, Board und Rules bilden die Spiellogik. Die Klasse Board hat zudem Zugriff auf die einzelnen Figure.
- Controller: Die Klasse CoreGame übernimmt die Kontrolle und vermittelt zwischen View und Model.

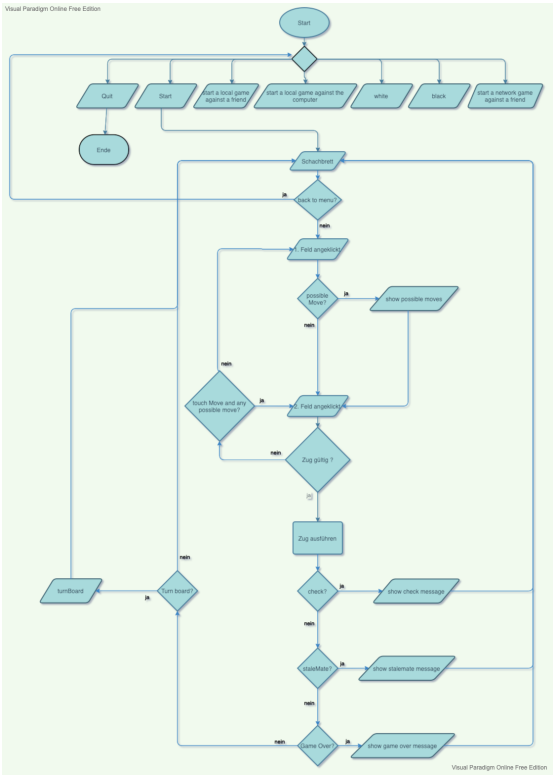
[1] Dieses Diagramm entspricht nicht eins zu eins den Regeln des UML-Klassendiagramms, sondern wurde so vereinfacht, dass nur die aussagekräftigen Methoden und Attribute dargestellt werden.

Durch diese Trennung der Komponenten ist es vergleichsweise einfach möglich, z.B. eine weitere View-Klasse hinzuzufügen, da die Spiellogik von der Ausgabe getrennt ist. Dadurch ist das Programm relativ unkompliziert erweiterbar und wartbar.

Dynamische Sicht

Der Programmablauf ist im Folgenden für das Spiel auf der Konsole sowie mit GUI skizziert:





[Programmablaufplan CLI]

[Programmablaufplan GUI]

Grundlegender Ablauf auf der Konsole:
Sobald das Spiel gestartet wird, wird in einer while-Schleife die Eingabe abgefragt und darauf reagiert. Bei einer Schachzug-Eingabe prüft das System, ob die Syntax korrekt war, dann ob der Zug semantisch gültig ist und führt anschließend den entsprechenden Zug aus. Das Spielfeld wird aktualisiert und das Spiel wartet auf eine erneute Eingabe des Spielers.

In der GUI ist die Ein- und Ausgabe entsprechend angepasst, die interne Logik bleibt aber dieselbe.