Projektdokumentation „HeckMeck“

Datum: 07.03.2014

Copyright © zetVisions AG 2014

Speyerer Straße 4

69115 Heidelberg

Germany

[www.zetvisions.de](http://www.zetvisions.de)

Inhalt

[Inhalt 2](#_Toc390854417)

[1. Vorprojektphase 3](#_Toc390854418)

[1.1 Aufgabenstellung 3](#_Toc390854419)

[1.2 Anforderungen 3](#_Toc390854420)

[2. Projektdefinition- und Planung 4](#_Toc390854421)

[2.1. Zeitliche Planung 4](#_Toc390854422)

[2.2. Spieldefinition 4](#_Toc390854423)

[3. Projektverlauf 5](#_Toc390854424)

[3.1 Objektmodell 5](#_Toc390854425)

[3.1.1 Grafische Benutzeroberfläche 5](#_Toc390854426)

[3.2 Spielablauf und -logik 6](#_Toc390854427)

[3.2.1. Beendigung eines Zuges 7](#_Toc390854428)

[3.2.2. Verursachung eines Fehlwurfs 7](#_Toc390854429)

[3.2.3. Sonderfall 7](#_Toc390854430)

[3.3 Message-System 8](#_Toc390854431)

[3.4 Mehrsprachigkeit 9](#_Toc390854432)

[3.4.1 Implementierung 9](#_Toc390854433)

1. Vorprojektphase

## 1.1 Aufgabenstellung

Es soll ein Spiel entworfen und entwickelt werden. Dabei liegen sowohl die Wahl der Programmiersprache, als auch alle weiteren Entscheidungen zur Umsetzung und Gestaltung bei dem ausführenden Team.

Außerdem ist neben einem Pflichtenheft auch eine vollständige Dokumentation anzufertigen und eine abschließende Präsentation über das Projekt vorzubereiten.

## 1.2 Anforderungen

Im Laufe des Projekts soll ein Spiel entstehen, welches sich an dem Würfelspiel „Heckmeck am Bratwurmeck“ orientiert. Das Spiel soll vollständig in Java programmiert werden und durch ein Server-Client-Modell mehreren Spielern über ein Netzwerk das Mitspielen erlauben. Dabei soll der Server den Großteil der Steuerung übernehmen und der Client möglichst unwissend bleiben. Dies soll eventuelle Eingriffe von außen verhindern oder zumindest schwieriger gestalten.

Des Weiteren soll das Spiel über eine intuitive grafische Oberfläche bedienbar sein. Hierbei soll kein GUI Builder verwendet werden, sondern die Oberfläche soll rein händisch implementiert werden.

Die Standardsprache sowohl für die Implementierung als auch das Spiel selbst soll Englisch sein.

2. Projektdefinition- und Planung

## 2.1. Zeitliche Planung

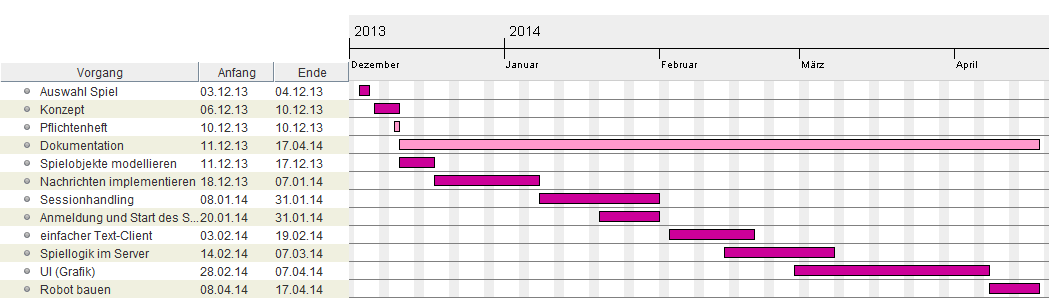


Abbildung 1

Anhand des Gantt-Diagramms sind sowohl die zeitliche Abfolge als auch der zeitliche Rahmen der einzelnen Arbeitsschritte gut zu erkennen.

Den größten zeitlichen Aufwand hat hier die Verwirklichung der grafischen Oberfläche. Darauf folgen die Implementierung von Nachrichten, Sessions und der Spiellogik. Diese Punkte sind auch als die Wichtigsten zu betrachten.

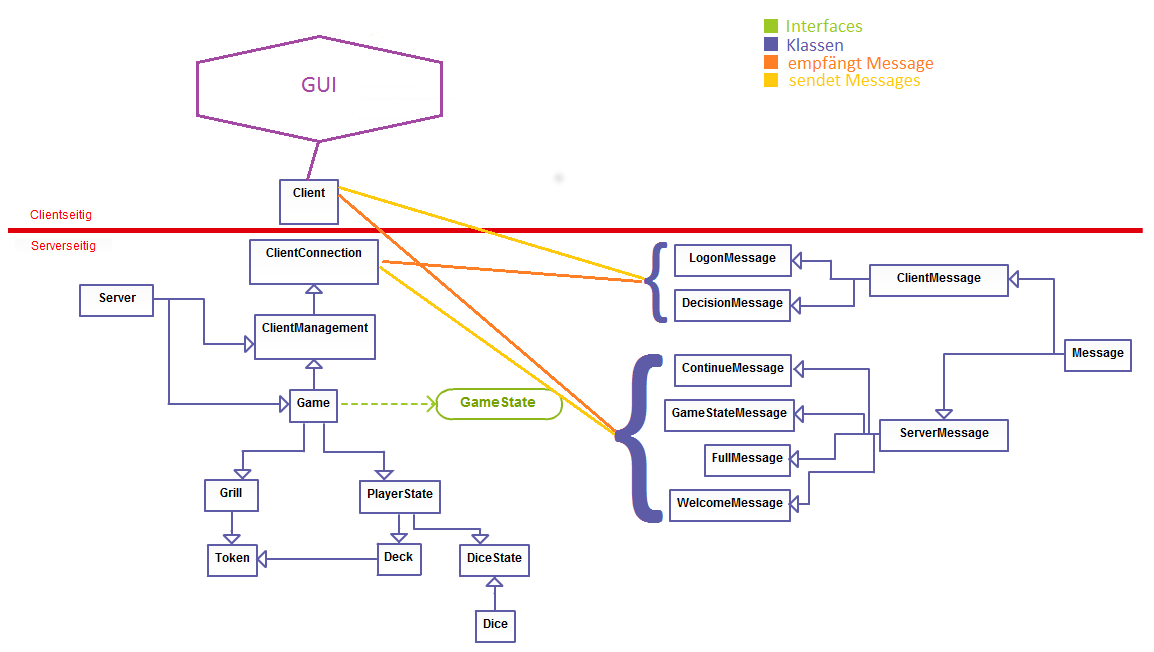
## 2.2. Spieldefinition

In der Spielmitte warten auf dem „Grill“ 16 unterschiedlich wertvolle Portionen „Bratwürmer“ darauf, von den würfelnden Hühnern ergattert zu werden. Dazu pickt ein Spieler identische Würfelsymbole aus seinem Wurf heraus und würfelt mit den verbliebenen Würfeln weiter. Erreichen seine Würfel den Wert einer ausliegenden „Bratwurmportion“, kann er diese nehmen und das sogar von den Vorräten seiner Mithühner.

Ziel des Spieles ist es, die meisten Bratwürmer zu sammeln.

3. Projektverlauf

## 3.1 Objektmodell



Abbildung

Im Laufe der Entwicklung des Spiels entwickelte sich auch das ursprüngliche Modell der Spielobjekte weiter und gewann deutlich an Komplexität.

Obiges Modell stellt nun das endgültige Modell der Objekte dar, wobei jedes Objekt als eigene Klasse zu verstehen ist. Die Kommunikation von Client und Server erfolgt hierbei über die Messageklassen.

Als zentrale Spielobjekte sind alle Klassen zu betrachten auf die Game Zugriff hat. Das sind im Wesentlichen der Grill mit Tokens, sowie der PlayerState welcher ein Deck aus Tokens sowie einen DiceState mit Dices enthält.

### 3.1.1 Grafische Benutzeroberfläche

Die grafische Oberfläche stellt den komplexesten Bereich der Entwicklung dar. Die Oberfläche wurde auch aufgrund der Übersichtlichkeit in mehrere Komponenten aufgeteilt.

Zum Starten des Servers wurde ein eigenes Fenster (GUIServer) implementiert auf welches ein Panel mit den nötigen Oberflächenelementen gelegt wird (GUIServerStart).

Das Spiel selbst wird über einen bestimmten Ablauf gerufen. Im Fenster (GUIClient) liegt standardmäßig die Anmeldeoberfläche (GUILogon). Mit dem Anmelden des Spielers wird diese ersetzt durch die Spieloberfläche, welche aus zwei Teilen besteht (GUIGame und GUIPlayer). GUIGame enthält hier das eigentliche Spiel, den so genannten Grill und die Würfel, während GUIPlayer je nach Anzahl der Spieler gefüllt wird und alle wichtigen Informationen zu diesen enthält.

Um das Spiel „auf einen Holztisch zu legen“ wurde die Klasse GUIBackground implementiert. Diese füllt sich selbst mit dem Holzpattern indem die paint-Methode redefiniert wurde und wird von allen anderen Klassen implementiert.

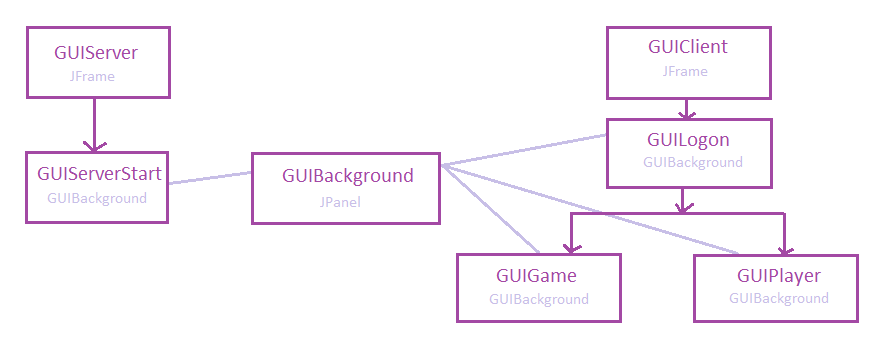


Abbildung - Aufbau Implementierung grafische Oberfläche

## 3.2 Spielablauf und -logik

Nach der Anmeldung der Spieler wird das Spiel wird durch den Server initialisiert. Das bedeutet, dass dieser die Spielsteine (Tokens) aufbaut sowie die Würfel (Dices) initialisiert und mit diesen einen Ausgangswurf erzeugt.



Abbildung - Auszug Spieloberfläche nach Initialisierung

Der zuletzt hinzugefügte Spieler beginnt dann das Spiel. Er würfelt und legt dann alle Würfel eines beliebigen Werts beiseite, z.B. alle Würfel, die eine 2 zeigen oder alle Würfel, die einen Wurm zeigen.

Mit den nicht beiseite gelegten Würfeln darf der Spieler erneut würfeln. Aus diesem Wurf legt er nun alle Würfel eines anderen Werts beiseite. Hat der Spieler beispielsweise im ersten Wurf alle 2er beiseite gelegt, so darf er nun keine 2er mehr dazu legen.

Dieses Würfeln und das anschließende Beiseitelegen von Würfeln darf der Spieler so lange fortsetzen, bis er entweder freiwillig seinen Zug beendet oder aber einen Fehlwurf verursacht.

### 3.2.1. Beendigung eines Zuges

Der Spieler darf seinen Zug jederzeit freiwillig beenden. Der Server zählt nun die Werte all seiner beiseite gelegten Würfel zusammen. Mindestens einer dieser Würfel muss einen Wurm zeigen – sonst handelt es sich um einen Fehlwurf.

Erreicht der Spieler mit seinen beiseite gelegten Würfeln genau den Zahlenwert einer Bratwurmportion, die auf dem Grill oder vor einem Spieler offen ausliegt, erhält er diese. Der Spielstein wird neben seinem Namen auf der Oberfläche abgelegt. Hat der Spieler bereits andere Spielsteine gesammelt wird ein Stapel simuliert und nur der oberste Spielstein wird angezeigt.

Erreicht der Spieler mit seinen beiseite gelegten Würfeln genau den Zahlenwert einer Bratwurmportion, die vor einem anderen Mitspieler ausliegt, wird diese gestohlen und vor ihm ausgelegt.

### 3.2.2. Verursachung eines Fehlwurfs

Es gibt Situationen durch welche der Spieler einen Fehlwurf auslösen kann.

1. Der Spiele konnte am Ende seines Zuges keine Bratwurmportion ergattern, weil der Wert der beiseite gelegten Würfel dazu nicht ausreicht oder weil er keinen Wurm beiseite gelegt hat.
2. Der Spieler würfelt in einem Wurf ausschließlich Zahlen/Würmer, die er bereits beiseite gelegt hat.

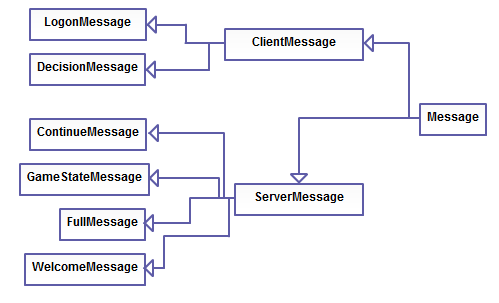
Damit hat der Spieler einen Fehlwurf verursacht und geht leer aus. Zudem muss er den oben liegenden Spielstein seines Stapels zurück auf den Grill legen. Der höchste dort ausliegende Spielstein wird außerdem umgedreht und somit aus dem Spiel genommen.

Danach ist der nächste Spieler an der Reihe.

### 3.2.3. Sonderfall

Liegt die Bratwurmportion, die man erwürfelt hat, weder auf dem Grill noch sichtbar vor einem Mitspieler, nimmt man die Portion mit dem nächstniedrigeren Wert vom Grill. Ist keine solche Portion mehr da hat der Spieler einen Fehlwurf verursacht.

## 3.3 Message-System

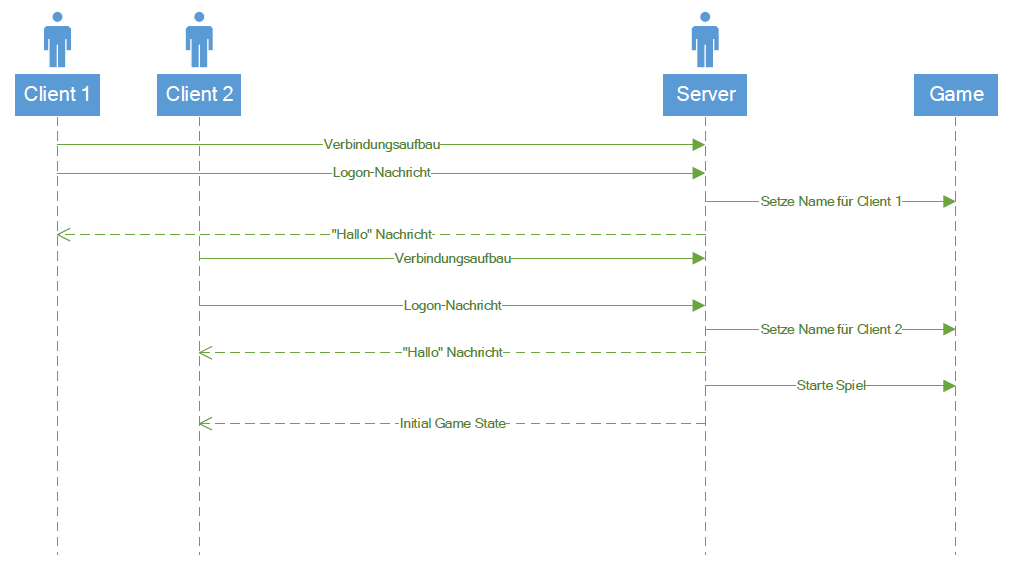


Abbildung

Die Verbindung zwischen Client und Server wird durch ein eigenes Messagesystem realisiert, welches die von Java bereitgestellte Architektur verwendet. Die Sockets implementieren hierbei die Client-Seite der Verbindung und der ServerSocket die Server-Seite der Verbindung.

Zwecks leichteren Handlings wurde das Messagesystem hierarchisch aufgebaut, wobei auch hier eine Serverseite sowie eine Clientseite herausgebildet werden.

An einem Beispiel des Anmeldeprozesses mit zwei Spielern kann dieses System noch einmal verdeutlicht werden:



Abbildung

## 3.4 Mehrsprachigkeit

Mit Hinblick auf Usability und Individualisierbarkeit wurde nach einer Möglichkeit gesucht, das programmierte Spiel weitestgehend mehrsprachig zu gestalten. Da die Programmier- und Hauptspielsprache Englisch ist, sollte als zweite Sprache Deutsch zur Auswahl stehen.



Abbildung 7 - Auszug Anmeldeoberfläche

Die Auswahl wurde über ein Dropdown auf dem Anmeldebildschirm realisiert. Dort kann der User nun wählen, ob er die Meldungstexte in Englisch oder Deutsch angezeigt bekommen möchte.

Anspruch an die Implementierung war, dass sie nicht zu kompliziert ist und gegebenenfalls um weitere Sprachen erweitert werden kann. Die Anforderungen konnten über eine Kapselung der Texte in einer neuen Klasse mit einem Resource Bundle realisiert werden.

### 3.4.1 Implementierung

Um die Funktionalität des Resource Bundles zu nutzen, müssen die Messagetexte in eigenen Dateien gekapselt werden. Diese Auslagerung in Dateien macht das System der Resource Bundles sehr leicht um weitere Sprachen erweiterbar.

Die Dateien müssen dem Muster Schlüssel = Text folgen, damit sie korrekt interpretiert und ausgelesen werden können.

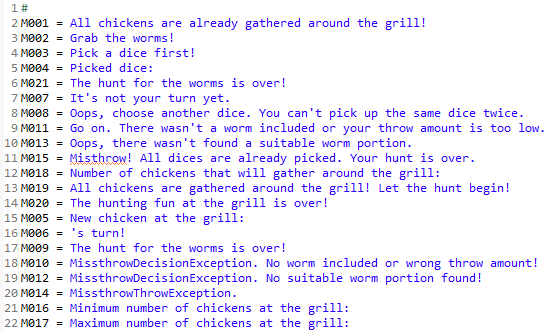


Abbildung 8- Beispieldatei englische Texte

Wird nun ein neues Spiel gestartet, sucht sich die Klasse anhand der gewählten Sprache die richtige Datei und liest aus dieser über den Schlüssel die entsprechenden Texte.

Durch diese Art der Umsetzung können erst die Texte zweisprachig erfasst werden, welche nach dem Klick auf „Logon“ (Anmeldeoberfläche) verwendet werden. Vorher wird die Standardsprache Englisch verwendet.