Quality Assurance

Wasserschlacht Simulator

Gruppe 04

Inhalt

[Einleitung 2](#_Toc483141097)

[Zuverlässigkeit 3](#_Toc483141098)

[CharacterTest 4](#_Toc483141099)

[ClientGameContollerTest 4](#_Toc483141100)

[GameMapTest 4](#_Toc483141101)

[PlayerTest 4](#_Toc483141102)

[ServerGameContollerTest 4](#_Toc483141103)

[TeamTest 4](#_Toc483141104)

[TileTest 4](#_Toc483141105)

[TileTypeTest 4](#_Toc483141106)

[TurnControllerTest 4](#_Toc483141107)

[WeaponTest 4](#_Toc483141108)

[WinningConditionTest 4](#_Toc483141109)

[WorldTest 5](#_Toc483141110)

[Effizienz 6](#_Toc483141111)

[Lessons learned 6](#_Toc483141112)

# Einleitung

Bereits am zweiten Meilenstein mussten wir unseren Tutoren ein Konzept für die Qualitätssicherung vorstellen. Damals hatten wir uns die Qualitätsmerkmale Funktionalität und Effizienz ausgesucht.

Wir wählten diese Merkmale, da sie ein reibungsloses Spiel versprachen.

Diese zwei Merkmale stellten wir dann auch am dritten Meilenstein vor dem Plenum vor.

Danach starteten wir mit den Unit-Tests für die Zuverlässigkeit und den Effizienzmessungen.

Für den letzten Meilenstein fassen wir nun alles zusammen.

# Zuverlässigkeit

Mit der Zuverlässigkeit als Qualitätsmerkmal wollen wir sicherstellen, dass das Spiel nicht abgebrochen wird. Denn uns ist es wichtig, dass die Spielperson während dem Spielen nicht von Bugs gestört wird. Wir hatten daher entschieden, die Spiel-Engine zu testen.

Die Anzahl der zu testeten Klassen war zu Beginn auf 50% der Engine angesetzt. Da wir unsere Engine besser testen wollten und viele Klassen essenziel sind, haben wir die Zahl erhöht. Wir sind nun bei 80% plus drei Klassen aus dem Package game. Zudem wollten wir, dass unsere Klassen gründlich getestet werden, weshalb eine code coverage von 90% angesetzt wurde.

Die Unit-Test werden nach dem Alphabet geordnet kurz angesprochen.

## CharacterTest

## ClientGameContollerTest

## GameMapTest

Mit der Klasse GameMap werden die Spielfelder kreiert, auf denen die Spielfiguren gesetzt werden. Es gibt zwei Hauptkarten und eine kleinere Testkarte. Die Testkarte wird nicht getestet. Die Karten werden beim Ausführen der Main von den Textdateien eingelesen. Auch bei dem Test werden im Setup alle Karten geladen. Zudem wird manuell eine dritte Karte mit einem char array erstellt.

Die im Setup ausgeführte einlese Methode wird durch einen Test geprüft, der die Existenz der Karten beweist. Alle restlichen Methoden werden in ihren Testmethoden aufgerufen und die Resultate der Ausführung mit Assert überprüft. Soweit es ging oder sinnvoll war, wurden alle Maps getestet. Zum Beispiel wurde eine Methode, die die Tiles einer Karte als char array zurückgibt, nur an der manuell erstellten Karte getestet, da das char array zur Erstellung als Vergleich diente konnte.

Nicht getestet wird die overriding Methode toString. Die code coverage war zu Beginn bei 20% und wurde im Laufe der Zeit auf 60%, 70% und schliesslich 90% erhöht. Die Klasse ist gründlich getestet und zeigte sich soweit fehlerfrei.

## PlayerTest

Hinter dem Player steht der User. Der Spieler wird automatisch einem Team hinzugefügt. Vom Player kann man den Namen, das Team, den User, die Farbe und Informationen bezüglich besiegten und verlorenen Spielfiguren erfahren. Für eine bessere Überprüfung der Ausführungsresultate wurden die CoreMatchers Methoden importiert. Diese Testmethoden bzw. Methoden funktionieren einwandfrei. Die Methoden zur Abfrage des momentanen Spielers und ob der Player noch Spielfiguren besitzt, funktionieren, konnten jedoch in der Testmethode nicht direkt vom Player-Objekt aufgerufen werden. Es handelt sich vermutlich um einen Fehler im Setup des Tests.

Die code coverage wurde von 60% auf 80% gesteigert. Nicht getestet wurden das Starten eines neuen Zuges und das Abfragen des Clients. Bei beiden Fällen liegt es am Setup des Testes.

## ServerGameContollerTest

## TeamTest

In unserem Spiel gibt es ein Player pro Team. Der Name des Teams ist im Spiel selbst nicht wählbar und wird automatisch erstellt. Im Test wird dem Testteam jedoch einen gewählten Namen gegeben. Der Name sollte für das Funktionieren der Methoden irrelevant sein. Mit einer Methode der Klasse, die alle Teammitglieder als ArrayList zurückgibt, wird nebenbei auch das erfolgreiche Hinzufügen des Testplayers bewiesen. Obwohl es nur ein Player pro Team gibt, testen wir das Hinzufügen in ein Team nochmals direkt mit einem zweiten Player. Die ArrayList enthält nun zwei Player, die Methode funktioniert und zeigt, dass die Erweiterung auf Teambildung möglich wäre. Der Test für die kleine Teamklasse erreicht eine code coverage von 100%.

## TileTest

## TileTypeTest

## TurnControllerTest

## WeaponTest

## WinningConditionTest

## WorldTest

# Effizienz

# Lessons learned