Sternenkrieg

Rebel Art Studios Inc.

Christian Bauer
chrbauer@edu.aau.at

Lukas Pagitz
l1pagitz@edu.aau.at

Anja Ressmann
anjare@edu.aau.at

Christoph Roither
chroither@edu.aau.at

Boda Wen
bodawe@edu.aau.at

21.06.2017

Software Engineering II - SS 16/17

Univ.-Prof. Dipl.-Ing. Dr. Martin Pinzger
Univ.-Ass. Dipl.-Ing. Christian Macho

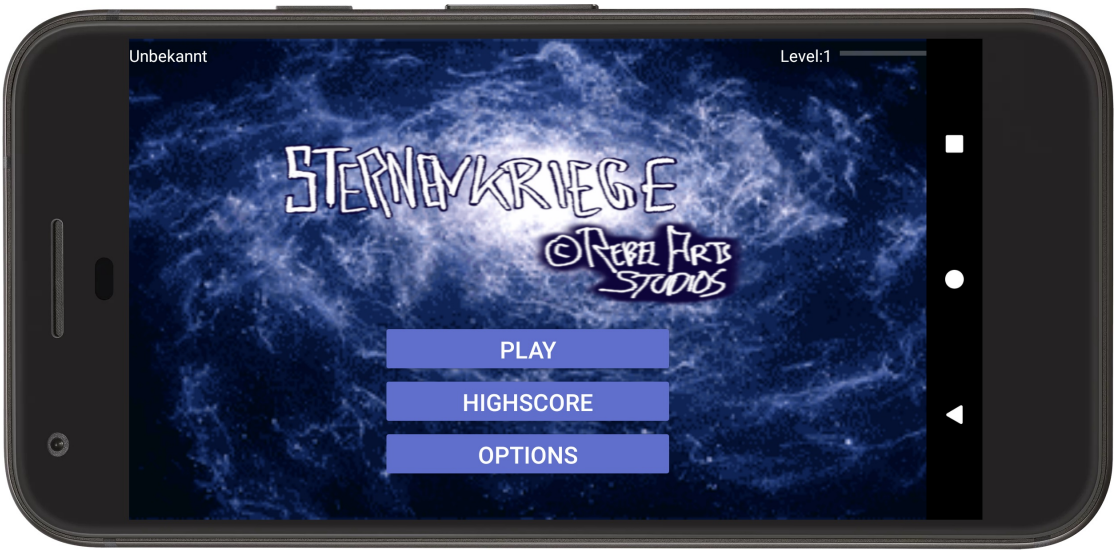

Abbildung 1: Startbildschirm von Sternenkrieg

Inhaltsverzeichnis

1 Einleitung

Das Spiel „Sternenkrieg“ baut auf dem Konzept des allgemein bekannten Brettspieles Schiffe versenken auf und verbindet dieses in moderner Art und Weise mit Weltall-Elementen.

Der Fokus des Projekts ist klar: Moderne Konzepte und ein interaktiver Spielablauf soll die Kunden dazu anregen, „Sternenkrieg“ gemeinsam am Smartphone zu spielen und sich dabei auch zu unterhalten.

Das wichtigste Key-Element des Spieles ist das sogenannte PowerUp-System: Dieses Feature bringt durch eine physische Würfelfunktion noch mehr taktische Tiefe in das Spiel. Die gewürfelte Augenzahl ermöglicht den Kauf von PowerUps, welche in jedem Spiel nur begrenzt verfügbar sind und eine besondere Funktion mit sich bringen, also möglicherweise in brenzligen Situationen den Spielverlauf umdrehen können. Auch die implementierte Schummelfunktion kann für einige Überraschung im Spiel sorgen.

2 Grundlegender Aufbau

2.1 Spielablauf

Sternenkrieg orientiert sich am Spielprinzip von „Schiffe versenken“. Ziel ist es, die vom Gegner platzierten Schiffe als erster Spieler zu zerstören. Als Erweiterung zum herkömmlichen Spiel, bietet Sternenkrieg weitere Features, deren technische Funktionalität hier kurz erläutert wird. Dieses Kapitel soll zum Verständnis des Ablaufs im Programm beitragen.

Gestartet wird das Spiel mit einem Klick auf den Button „Play“. Danach muss eine Verbindung zwischen beiden Spielern hergestellt werden. Hierzu ist es notwendig, dass sich die Smartphones beider Spieler im gleichen WLAN-Netzwerk befinden. Ein Spieler übernimmt im Spiel den Server, der andere übernimmt den Client.

Der Server-Spieler startet danach den Server und bekommt eine IP-Adresse angezeigt, welche er dem Client-Spieler übermittelt. Dieser verbindet sich damit zum Server. Sobald die Verbindung erfolgreich war, melden sich beide Spieler als bereit und das Spiel beginnt.

Um zu entscheiden, welcher der beiden Spieler das Spiel beginnen soll, muss gewürfelt werden. Dies geschieht durch Schütteln des Smartphones. Der Wert des jeweiligen Gegners wird dabei über das Netzwerk übertragen.

Nun beginnt das eigentliche Spiel: Jeder Spieler muss seine Schiffe (ein kleines, ein mittleres und ein großes) platzieren. Ziel des Gegners ist es nun, die Schiffe des Gegners zu zerstören, bevor die eigenen vernichtet werden.

Im weiteren Spielverlauf werden auch die besonderen Features von Sternenkrieg sichtbar. Diese sind unter anderem eine Schummelfunktion sowie ein PowerUp-System.

2.2 Menüaufbau

Das Menü von Sternenkrieg bietet - abgesehen vom eigentlichen Spiel - noch folgende Buttons:

- Highscore – zeigt den Highscore von Sternenkrieg an
- Options – öffnet die Einstellungen des Spiels
- About – zeigt eine kurze About Us-Meldung an

3 Aufbau des Projekts

3.1 Aktivitätsdiagramm

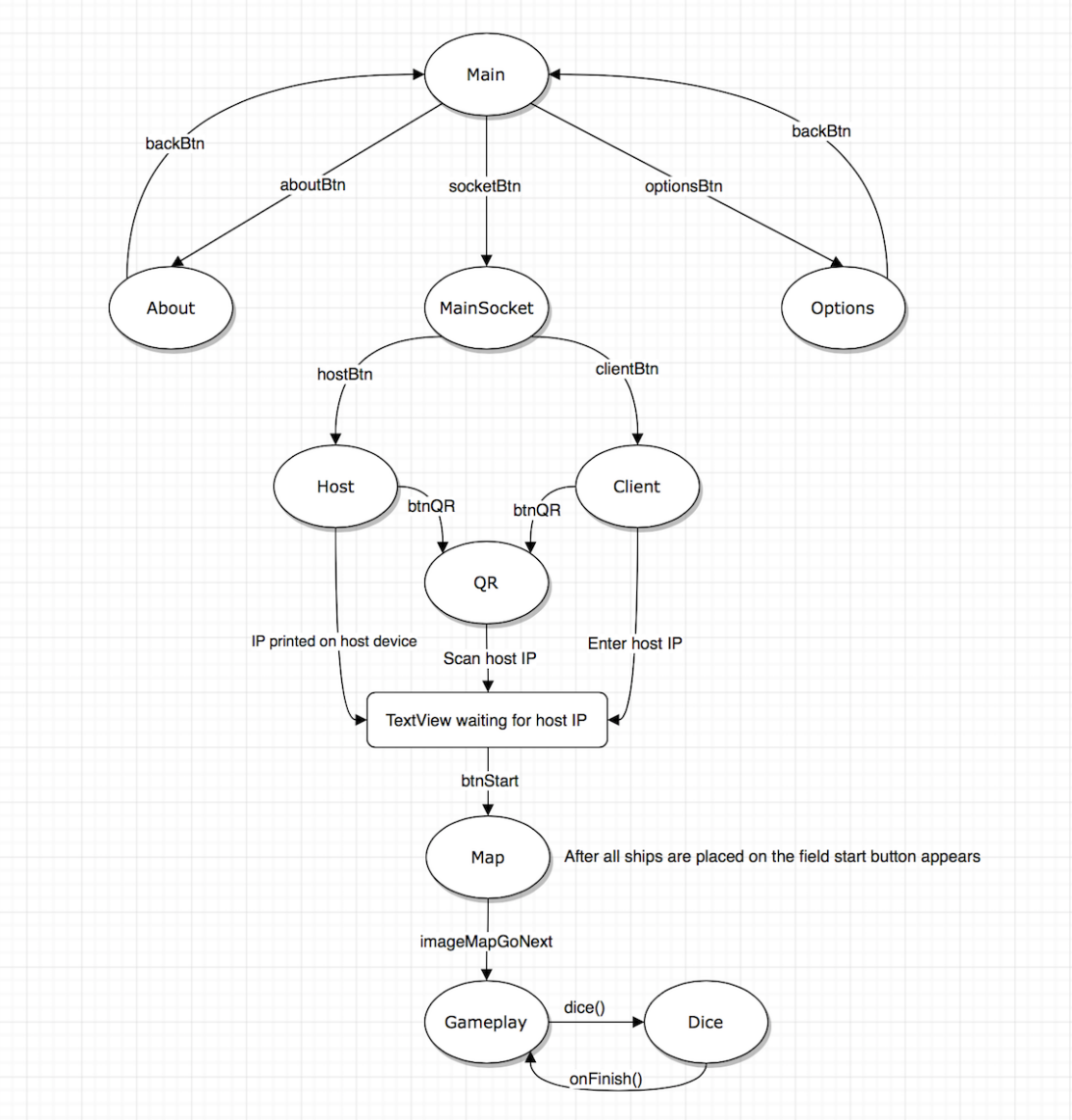

Abbildung 2: Aktivitätsdiagramm

3.2 Programmaufbau

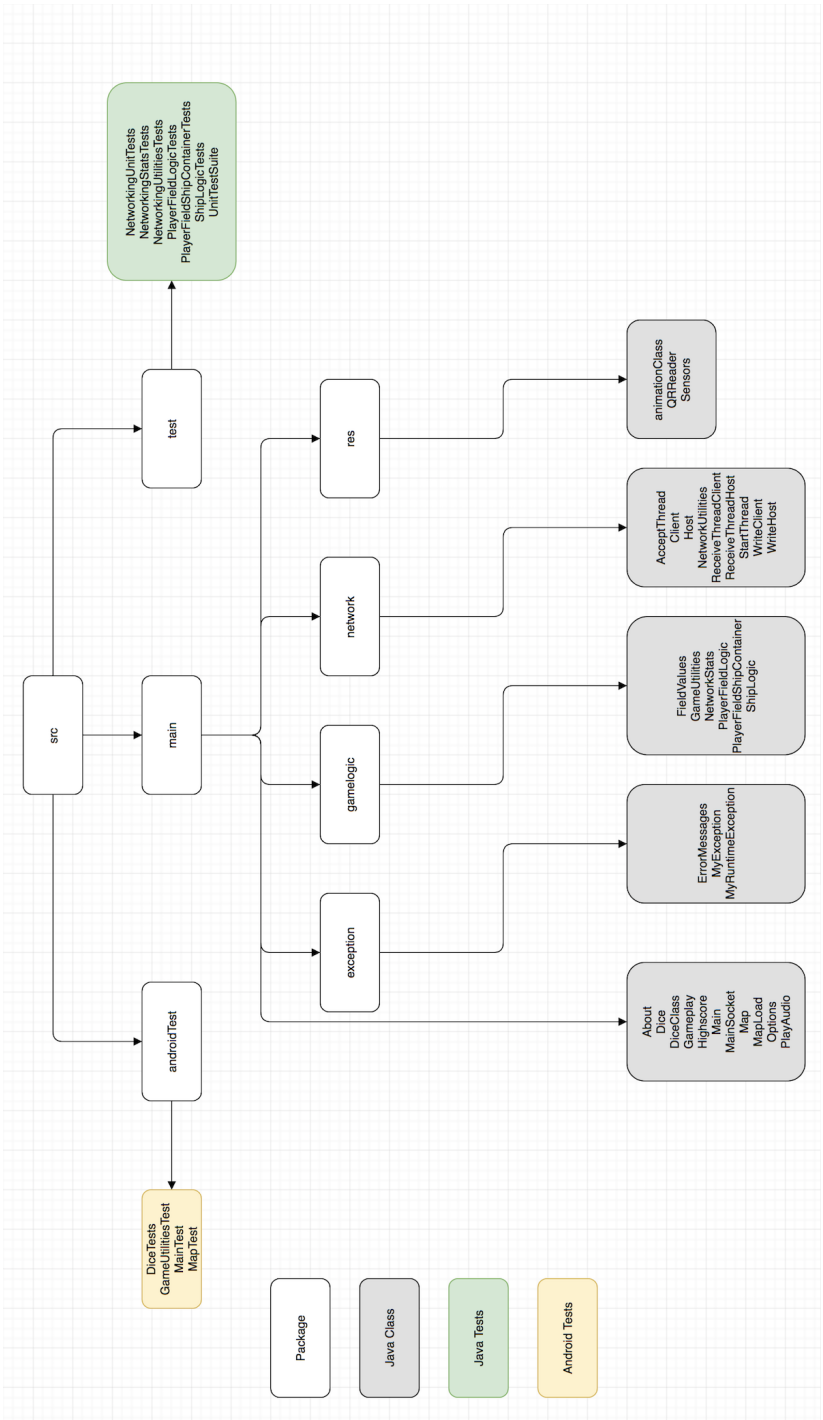

Abbildung 3: Programmaufbau (Klassenübersicht)

4 Klassen im Überblick

In diesem Kapitel werden die allgemeinen Klassen und die dazugehörigen Activities von Sternenkrieg beschrieben. Besonders wichtige Klassen und Methoden werden ausführlicher behandelt.

4.1 About

Die Klasse „About“ gibt einen kurzen Text über Sternenkrieg aus und hat sonst keine weitere Funktionalitäten.

4.2 Dice

Die Klasse „Dice“ steht einen Würfel zur Verfügung. Dieser Würfel wird am Beginn des Spiels verwendet um den Beginner des Spiels zu bestimmen. Der Würfel wird außerdem bei den PowerUps verwendet, um eine Zufallszahl zu erhalten. Das Würfeln erfolgt durch Schütteln des Smartphones. Hierfür wird ein SensorEvent-Listener verwendet. Unterstützt wird die Klasse Dice durch die Klasse res/Sensors.
Die Auswahl des korrekten Würfels (Spielstart oder PowerUps) erfolgt durch die Auswahl eines Modus (Variable mode). mode = 1 steht hierbei für den Spielstart, mode = 2 für das PowerUp-System.

4.3 DiceClass

Die Klasse „DiceClass“ unterstützt den Würfel mit einigen ausgelagerten Methoden (Wechseln der Bilder und Abrufen der Variablen).

4.4 Gameplay

In Klasse „Gameplay“ passiert das eigentliche Spiel. Durch die Netzwerkfunktionen im Hintergrund wird mit dem Gegenspieler kommuniziert. In dieser Klasse wird vor allem mit onClickListener() gearbeitet. Wird auf das gegnerische Feld geklickt, so wird dies an den Gegenspieler übermittelt. Ebenso werden hier die Punkte für den Highscore ermittelt. Der SensorEventListener holt sich den aktuellen Helligkeitswert der Umgebung und je nach Wert wird ein bestimmtes Hintergrundbild festgelegt. Ist es dunkel, so leuchten weniger Sterne, bei heller Umgebung leuchten mehr. Auch die Cheat-Funktion und die PowerUps befinden sich in der Gameplay-Klasse. Diese werden ebenfalls durch onClickListener realisiert: Positionen und Werte der einzelnen Schiffe werden korrekt berechnet und and den Gegenspieler übetragen. Zusätzlich steht eine Chat-Funktion zur Kommunikation mit dem Gegenspieler zur Verfügung.

4.4.1 Auswahlmöglichkeiten

Das Spielfeld bietet folgende Buttons zur Auswahl:

- Cheat-Funktion

- PowerUps
- Optionen
- Hilfe

Wird die PowerUp-Funktion ausgewählt, so verändern sich die Icons in PowerUp-Icons, damit eine Funktion ausgewählt werden kann.

4.4.2 Cheat-Funktion

Die Cheat-Funktion im Spiel erlaubt es, ein noch nicht angegriffenes Schiff auszuwählen und dieses an einer anderen Stelle neu zu platzieren. Danach wird der eigene Zug fortgesetzt.

4.4.3 PowerUps

PowerUps sind Funktionen, die einem Spieler einen bestimmten Vorteil verschaffen sollen. Am Anfang jedes Zuges wird gewürfelt, diese Punkte werden zu den PowerUp-Punkten dazugezählt und können anschließend für ein PowerUp verwendet werden. Die PowerUps bieten folgende Funktionen:

- doppelte Rüstung für ein noch nicht angegriffenes Schiff (das große Schiff müsste statt drei Mal nach Aktivierung dieses PowerUps sechs Mal aktiviert werden)
- im aktuellen Spielzug darf zwei Mal auf das Spielfeld des Gegners geschossen werden

Besitzt ein Spieler zu wenig Punkte oder wurde ein PowerUp bereits zu oft verwendet, wird eine entsprechende Meldung ausgegeben.

4.5 Highscore

Die Klasse „Highscore“ zeigt den Highscore des Spiels an. Die Daten werden dabei aus der Klasse GameUtilities geholt.

4.6 Main

Die Klasse „Main“ stellt den Startscreen sowie das Hauptmenü dar. Eingebunden werden hier das Hintergrundbild sowie die Buttons zur Menüauswahl. Falls das Spiel zum ersten Mal gestartet wird, muss vom Spieler ein Name eingegeben werden.

4.7 MainSocket

Die Klasse „MainSocket“ wird nach Klick auf „Play“ im Hauptmenü aufgerufen. In diesem Menü kann ein neues Spiel erstellt („Server“) bzw. einem Spiel beigetreten („Client“) werden. Der Server-Spieler bekommt eine IP-Adresse angezeigt, welche er dem Client-Spieler übermittelt. Dieser gibt diese IP-Adresse ein und verbindet sich damit zum Server.

4.8 Map

Die Klasse „Map“ stellt das eigene Spielfeld dar, welches nach erfolgreichem Würfeln erscheint. Hier werden per Drag&Drop die drei Raumschiffe platziert. Mithilfe der Klasse MapLoad wird ein Grid initalisiert und somit werden die einzelnen Felder richtig angezeigt. Die Drag&Drop-Funktion wird durch einen setOnDragListener realisiert. Außerdem können Schiffe mit einem Button gedreht werden. Die Map-Klasse berechnet nun welche Felder von einem gedrehten Schiff eingenommen werden. Erst wenn alle drei Schiffe platziert wurden, kann per Button zum nächsten Screen und somit zum eigentlichen Spiel gewechselt werden.

4.9 MapLoad

Die Klasse „MapLoad“ wird aufgerufen, wenn ein neues Spielfeld-Grid (Gameplay) initialisiert wird. In dieser Klasse wird hauptsächlich festgelegt, wie die Felder von beiden Spielern dargestellt werden. Die Größe des Grids wird in der getView-Methode durch die Bildschirmgröße des Geräts ermittelt und angepasst. Die wichtigste Methode ist setFieldColor. Darin wird aufgrund des Strings im String-Array (jeder Eintrag wird einem Feld im Grid zugeordnet), welcher einen gewissen Status des jeweiligen Feldes angibt, die richtige Farbe bzw. das richtige Schiffbild zugewiesen.

4.10 Options

Die Klasse „Options“ behandelt alle Einstellungsmöglichkeiten von Sternenkrieg. Dazu zählen:

- Name
- Sound an/aus
- Sprache (DE/EN)

Die Speicherung der Einstellungen erfolgt mithilfe von SharedPreferences („prefs“). Diese bieten eine einfache Speicherung von Variablen, welche ebenfalls sehr einfach aktualisiert werden können. Nach einer Änderung wird die entsprechende Variable neu abgespeichert und steht somit auch den anderen Klassen zur Verfügung.

4.11 PlayAudio

Die Klasse „PlayAudio“ dient zum Abspielen der Hintergrundmusik.

5 GameLogic-Klassen im Überblick

In diesem Kapitel werden die GameLogic-Klassen näher erläutert. Wie der Name schon sagt, beinhalten diese Klassen die eigentliche Spiellogik.

5.1 FieldValues

Die Klasse „FieldValues“ ist eine unterstützende Hilfeklasse der Klassen Gameplay, Map und MapLoad. In dieser Klasse werden Informationen über die Schiffe sowie die Schiffe in Listen gespeichert.

5.2 GameUtilities

Die Klasse „GameUtilities“ verwaltet das Speichern der Informationen zum Spiel, darunter:

- Fortschritt
- Highscore
- Level
- Sound an/aus
- Username

Diese Informationen werden in den Shared Preferences „prefs“ gespeichert.
Die Klasse bietet unter anderem folgende Methoden:

- ArrayList¡String¿ getHighscore() - liefert den aktuellen Highscore
- void setHighscore() - setzt den aktuellen Highscore
- void deleteHighscore() - löscht den Highscore
- String getUsername() - liefert den gespeicherten Namen des Spielers
- void setUsername(String username) - setzt einen neuen Namen (beim Start des Spiels bzw. in den Optionen)

5.3 NetworkStats

In der Klasse „NetworkStats“ werden Variablen für das Netzwerk sowie der Würfelmodus gespeichert. Diese Variablen werden zur Verbindung genutzt, zusätzlich wird die Würfeleinstellung für den Würfel zur Unterscheidung zwischen Spielstart und PowerUp verwendet.

5.4 PlayerFieldLogic

In der Klasse „PlayerFieldLogic“ wird die Logik des Spielfelds verwaltet. Die Spielfelddaten werden in einem String-Array playerField gespeichert.
Ähnlich wie die ShipLogic-Klasse besitzt auch diese Klasse mehrere Methoden, um die einzelnen Schiffsgrößen auf dem Spielfeld zu platzieren. Dies geschieht durch Übergabe der Position, des Winkels und eines Strings zur. Dabei wird bei den einzelnen Parametern überprüft, ob die Position der zu setzenden Felder gültig ist, ob der Winkel entweder horizontal oder vertikal ist und ob der String gültig ist.

5.4.1 ContainsString

Für die verschiedenen Schiffsgrößen existieren eigene ContrainsString-Methoden. Dabei werden Methoden aus der Klasse ShipLogic zur Überprüfung verwendet (inRange, getSibling, String != null).

5.5 PlayerFieldShipContainer

Um den Workflow und die Lesbarkeit zu erhöhen, werden die Klassen ShipLogic und PlayerFieldLogic in der Klasse „PlayerFieldShipContainer“ zusammengeführt, um Arbeitsschritte zu verringern. Der Konstruktor beinhaltet genau diese beiden Klassen.
Es existieren drei setShipContainer-Methoden (für die drei Größen), wobei das jeweilige Schiff in beiden Klassen gesetzt wird.

5.6 ShipLogic

Die Klasse „ShipLogic“ koopieriert mit der Klasse PlayerFieldLogic. In der Klasse ShipLogic werden die Schiffspositionen abgespeichert und bei Bedarf (z.B. beim Löschen der Schiffspositionen auf dem Spielfeld) abgerufen.

5.6.1 Positionen der Schiffe

Die Positionen der Schiffe werden in diesen Variablen gespeichert:

- int[] smallShip
- int[] middleShip
- int[] bigShip

Für alle Arrays stehen entsprechende get-Methoden zur Verfügung.
Die Position jedes Schiffes kann entweder über das Array gesetzt oder direkt als Position mit horizontalem/vertikalem Winkel angegeben werden. Der Winkel wird mit der Methode getSibling(int position) aufgerufen und übergibt den Abstand zu der Position.

5.6.2 getSibling

Die Methode int getSibling(int position) überprüft den übergebenen Parameter und liefert als Rückgabewert entweder 1 oder 8, welcher für die Positionierung der Siblings verwendet wird. Bei horizontaler Ausrichtung werden diese um eine Position versetzt auf das Spielfeld gesetzt, bei vertikaler Ausrichtung um 8 Positionen versetzt positioniert. Bei middleShip und bigShip wird stets mit dem Vorgänger-Sibling begonnen (position - sibling-Abstand bzw. position + sibling-Abstand).

5.6.3 ShipIsSetOnField

Schiffe können per Drag&Drop auf dem Spielfeld platziert werden. Es existieren daher drei boolean-Variablen, welche angeben, ob ein Schiff bereits positioniert wurde oder nicht. Um zu prüfen, ob bereits alle Schiffe gesetzt wurden, steht die Methode allShipsSetOnPlayerField() zur Verfügung. Falls diese Methode true als Rückgabewert liefert, wird das ImageView in der Map-Activity auf aktiviert gesetzt und der Spieler kann fortfahren.

5.6.4 inRange

Diese Methode prüft, ob sich die übergegebenen Positionen innerhalb des Spielfelds befinden.

6 Network-Klassen im Überblick

Alle Netzwerk-bezogenen Klassen befinden sich im Package „network“. Im Spiel Sternenkrieg übernimmt ein Spieler die Rolle des Servers, der andere die des Clients. Es wird sozusagen ein Server erstellt, zu welchem sich der Client verbindet. Im weiteren Spielverlauf spielt dies keine Rolle.

6.1 AcceptThread

Die Klasse „AcceptThread“ stellt einen Thread für die Socket-Kommunikation zur Verfügung. In dieser Klasse wird der SocketServer erstellt (siehe StartThread für Socket).

6.2 Client

Die Klasse „Client“ wird vom Client-Spieler geöffnet. Dieser verbindet sich mithilfe eines Threads per Socket zum Server. Wurde eine Verbindung hergestellt, kann der Server-Spieler das Spiel starten.

6.3 Host

Die Klasse „Host“ wird vom Server-Spieler geöffnet. Dieser öffnet einen Socket (per Thread) und wartet anschließend auf die Verbindung vom Client. Die eigentlichen Spieldaten werden in der Klasse GameUtilities gespeichert. Wurde die Verbindung erfolgreich hergestellt und das Spiel manuell vom Host gestartet, beginnt das Würfeln für den Spielstart.

6.4 NetworkUtilities

Die Klasse „NetworkUtilities“ stellt die zentrale Schnittstelle für das Netzwerk dar und verbindet alle anderen Netzwerk-Klassen miteinander

6.5 ReceiveThreadClient/ReceiveThreadHost

Die Klassen „ReceiveThreadClient“ bzw. „ReceiveThreadHost“ dienen zum Empfangen von Nachrichten über den Socket. Eine Klasse dient dabei für den Server-Spieler, die andere für den Client-Spieler. Write-Klassen: siehe WriteClient/WriteHost.

6.6 StartThread

Die Klasse „StartThread“ stellt einen Thread für die Socket-Kommunikation zur Verfügung. In dieser Klasse wird der eigentliche Socket erstellt (siehe AcceptThread für SocketServer).

6.7 WriteClient/WriteHost

Die Klassen „WriteClient“ bzw. „WriteHost“ dienen zum Senden von Text an den Client-Spieler bzw. an den Server-Spieler. Dabei wird die Nachricht über den entsprechenden Socket verschickt. Receive-Klassen: siehe ReceiveThreadClient/ReceiveThreadHost.

7 Resources-Klassen im Überblick

7.1 Animation

Die Klasse „Animation“ ermöglicht das Animieren eines Buttons (Wechseln zwischen zwei Farben). Diese Funktion wird beim Erstellen eines Spiels verwendet.

7.2 QRReader

Die Klasse „QRReader“ stellt einen QR-Reader bereit. Dieser kann beim Erstellen eines Spiels zum Austausch der IP-Adresse verwendet werden (als Alternative zur manuellen Eingabe).

7.3 Sensors

Die Klasse „Sensors“ ist eine unterstützende Hilfsklasse für die Klasse Dice und beinhaltet Berechnungen zur Bewegungserkennung des Smartphones.

8 Exception-Klassen im Überblick

Es wurden eigene Exceptions eingefügt, um Meldungen im SonarQube zu beheben.

8.1 ErrorMessages

Die Klasse „ErrorMessages“ verwaltet zentrale Fehlermeldungen. In dieser Klasse werden nur Strings gespeichert.

8.2 MyException

Die Klasse „MyException“ ist eine Erweiterung der Klasse Exception, stellt aber keine weiteren Funktionen zur Verfügung.

8.3 MyRuntimeException

Die Klasse „MyRuntimeException“ ist eine Erweiterung der Klasse RuntimeException, stellt aber keine weiteren Funktionen zur Verfügung.

9 Grafiken

Für das Spiel wurden verschiedenste Grafiken erstellt. Diese wurde in den entsprechenden res-Verzeichnissen des Projekts abgelegt (u.a. Schiffe, Hintergrundbilder, Würfel, ...).

10 Backlog

Funktion	Beschreibung	Erledigt
Anmeldefunktion	Nach dem Start der Anwendung muss man einen Spielnamen eingeben, diesen kann man später in den Optionen ändern.	ja
Optionen	In den Optionen kann man den Anzeigenamen sowie die Musik (ein/aus) einstellen sowie das Leaderboard einsehen.	ja
Multiplayerfunktion	Das Spiel beinhaltet eine Multiplayerfunktion, die durch 2 Spieler über WLAN initialisiert wird.	ja
Chat	Die Chatfunktion beinhaltet den Textaustausch mit den Mitspielern.	ja
Drag&Drop	Startet man ein Spiel, so muss man die Schiffe durch Drag&Drop auf das Spielfeld setzten.	ja
Würfelfunktion	Der Würfel kann durch ein einfaches Schütteln des Handys gewürfelt werden, dies wird für die Startreihenfolge sowie für die PowerUps benötigt.	ja
PowerUps	PowerUps sind ein Key-Element des Spieles. Diese kann man durch Bezahlen von Würfelpunkten aktivieren und somit einen strategischen Vorteil erzielen.	ja
Schummelfunktion	Diese Funktion beinhaltet das Verschieben eines Schiffes an einen beliebigen Punkt.	ja
Interaktive Map	Die Spielfeldkarte kann durch Hilfe des Lichtsensors an die Umgebung angepasst werden. Das heißt: Spielt der Nutzer unter starkem Sonnenlichteinfall, so kann die Karte einen besseren Kontrast zu den Schiffen und anderen Elementen erzeugen. Spielt der Nutzer in einer dunkleren Umgebung, so könnte die Spielfeldkarte schwarz sein und pulsierende Sterne beinhalten.	ja
Leaderboard	Hier werden lokal alle Spiele mit den erreichten Punkten angezeigt.	ja