Software Engineering 1
Abgabedokument
Teilaufgabe 2
(ReadMe)

Nachname, Vorname:	Srdanovic, Nemanja
Matrikelnummer:	01576891
E-Mail-Adresse:	a01576891@unet.univie.ac.at
Datum:	11.12.2020

Unteraufgabe 2: ReadMe für Teilaufgabe 2

Algorithmus Kartengenerierung

Der Algorithmus für die Kartengernerierung befindet sich im client.map package und beinhaltet die Klassen MapController die eine Karte mit fix vorgegebener Anzahl an Terrain Typen (Angabe aus der Spielidee) generiert. Diese Angaben befinden sich in dem client.constants package bzw. in der GameConstants Klasse, wodurch gewährleistet wird, dass sich die Generierung einfach anpassen kann, sollte sich etwas an den Anforderungen ändern.

Alle Felder der Karte werden zufällig aufgestellt, nur die Anzahl der Terrain Typen ist vorgegeben. Durch die Klasse MapValidator wird gewährleistet, dass alle Anforderungen bezüglich Inseln, Grenzen etc. erfüllt werden. Dies wird wie vorgeschlagen durch den Floodfill Algorithmus (fieldAccessibilityRule) sowie eine Anzahl von anderen hauptsächlich aus Schleifen bestehenden Algorithmen gewährleistet.

Die Kartengenerierung ist so optimiert, dass automatisch vom MapGenerator karten zufällig generier werden, solange eine Karte nicht alle Anforderungen erfüllt. In der Regel passiert dieser Vorgang sehr schnell.

Algorithmus zur Wegfindung

Für die Wegfindung wird wie empfohlen der Djikstra Algorithmus verwendet, um den Ansatz zur Erstellung bzw. eine Idee zu bekommen, habe ich mir die Pseudocode´s und die Beschreibungen von vollenden Seiten angeschaut (Bei jeder Seite bis zum Teil Implementierung):

- Wikipedia: (https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm)
- Softwaretestinghelp: (https://www.softwaretestinghelp.com/dijkstras-algorithm-in-java/)
- Stackabuse: (https://stackabuse.com/graphs-in-java-dijkstras-algorithm/)

Um die Beste bzw. kürzeste Rute auszuwählen, wird zuerst ein selbsterstellter Algorithmus (eine Art BTS) verwendet. Dabei werden alle unbesuchten Felder in der Umgebung der Position, welche Grass Felder sind, in eine Liste gespeichert. Es wird mit den unmittelbaren Nachbaren begonnen, falls diese schon alle besucht sind oder keiner der Felder ein Grass Feld ist, werden die Nachbaren der Nachbaren genommen usw. bis alle unbesuchten Grass Felder, die sich in der Umgebung befinden, ausgewählt werden. Falls z.B. alle Felder in der Umgebung besucht sind, dehnt sich der Alg. wie eine Art Floodfill aus, bis er unbesuchte Felder findet.

Wenn unbesuchte Felder gefunden worden, wird die Liste mit den unbesuchten Feldern, an den RouteCalculator übergeben, der mithilfe des Djikstra Alg. für jedes dieser Felder die kürzeste Rute berechnet. Aus diesen berechneten Ruten wird, die kürzeste ausgewählt und an die AI zurückgesendet, damit die AI Anweisungen zur Durchquerung dieser Route senden kann.

Netzwerk und Converter

Die Netzwerk und Converter Klassen sind in packages client.converter bzw. client.network aufgeteilt. Die Kombination dieser beider Klassen geschieht in der Klasse GameController bzw. GameStateWorker. Objekte aus diesen zwei Klassen werden bevor sie an die Netzwerkklasse weitergeleitet werden, an die Converter Klasse überreicht, um in Network passende Objekte umgewandelt zu werden.

MVC

Die MVC Bestandteile sind in den Klassen GameController (Controller) , Game(Model) und Cli(View) umgesetzt. Der GameController ändert, nachdem die Daten von der Netzwerkklasse über den Konverter konvertiert sind, die Daten in der Game Klasse, wodurch das firePropertyChange getriggert wird und die neuen Daten über den Clie angepasst und ausgegeben werden.