BENUTZERHANDBUCH, *Version: 1.0* - TEAM 08
SOFTWAREGRUNDPROJEKT WISE 2021/2022 UND SOSE 2022

BEARBEITET VON

LEONHARD ALKEWITZ, LOUIS BOCK, ALEXANDER FINK,
JANNIS KIESELBACH, FELIX SCHOPPE, NIKLAS SCHUERRLE

Inhaltsverzeichnis

1 Einführung

In diesem Projekt geht es um die Entwicklung des Multiplayer-Spiels „Deserts of Dune“. Das Projekt beinhaltet die Planung als auch Implementierung der vom Stakeholder vorgegebenen Anforderungen und Wünsche. Die Teammitglieder sollen durch die Umsetzung des Projektes lernen, praktische Erfahrungen mit den Methoden der Lehrveranstaltung zu sammeln. Zum Abschluss des Projektes wird dem Kunden ein funktionstüchtiges Spiel ausgeliefert, welches den Anforderungen entspricht.

1.1 Motivation

Die Durchführung des Projektes findet im Rahmen des Informatik-Studiums statt. Die Studenten sollen hierbei das Projekt planen und implementieren. Hierbei sollen die Methoden der Lehrveranstaltung angewandt werden.

2 Der Server

2.1 Allgemeines

Dieser Abschnitt behandelt die Handhabung des Servers, welcher von Team 08 im Rahmen des Softwaregrundprojekt im WISE 2021/2022 und SOSE 2022 an der Universität Ulm entwickelt wurde.

2.2 Installation und Einrichtung

Eine Voraussetzung für die Ausführung des Servers ist das Vorliegen der aktuellen Docker Version.

2.2.1 Bauen und Starten des Docker-Image

Für den Container liegt ein Dockerfile vor. Mithilfe eines Shell kann die docker-compose.yml Datei ausgeführt werden. Das docker-compose.yml Skript übernimmt das Bauen des Images für den Server. Der Docker-Container kann nun ohne weiteres gestartet werden. Voraussetzung dafür ist, dass Ordnerstruktur exakt übernommen wird und es den Ordner *Server* und *GameData* gibt und in deren gemeinsamen Elternverzeichnnis die Dockerfile und docker-compose.yml liegen.

2.2.2 Kompilieren des Programms

Es ist auch möglich, das Projekt manuell zu bauen und auszuführen. Dafür muss der Server mit *dotnet build Server.csproj -c Release* gebaut werden und dann mit *dotnet publish Server.csproj -c Release* gepublished werden. Dann sollte ein Ordner erstellt werden (für gewöhnlich bin/Debug/...), der die .dll Dateien enthält, die notwendig sind, um das Projekt auszuführen. Wechselt man nun in den Ordner und führt *dotnet Server.dll <argumente>* aus, um den Server zu starten.

2.2.3 Kommandozeilenoptionen des Servers

Der Server besitzt alle im Standard aufgelisteten Konfigurationsmöglichkeiten(auch optionale Kommandozeilenoptionen). Weitere Kommandozeilenoptionen sind nicht vorgesehen.

Auflistung der Kommandozeilenoptionen des Servers:

config-party: Fordert einen absoluten oder relativen Pfad zur .party.json

config-scenario: Fordert einen absoluten oder relativen Pfad zur <Name>.scenario.json

x: Ermöglicht es optionale Schlüssel-Wert-Paare zu übermitteln, die verfügbaren Einstellungen sind von Server zu Server individuell.

help: Listet alle verfügbaren Parameter auf. Falls individuelle Parameter hinzugefügt wurden so ist dieser Parameter verpflichtend.

port: Fordert nachfolgend eine (gültige) Portnummer, wird keine angegeben oder sollte dieser Parameter nicht unterstützt werden, so wird die Portnummer 10191 ausgewählt.

2.2.4 Architektur und Funktionsweise des Servers

Die Projektmappe des Servers besteht aus einem **GameData** Modul und einem **Server** Modul. Das GameData Modul beinhaltet unter anderem ein Netzwerkmodul, welches für die Serialisierung und Deserialisierung von Nachrichten verantwortlich ist. Zudem kümmert sich das Netzwerkmodul um die Konnektivität von Client und Server. Außerdem beinhaltet das Netzwerkmodul einen Ordner util, welcher wichtige Spieldaten, die sowohl für den Client als auch für den Server relevant sind, beinhaltet. Zu diesen gehören beispielsweise Charaktere, die Karte und die GreatHouses. Hier liegt zudem ein Logger vor. In GameData liegt ein Ordner Configuration vor, welcher für die Konfiguration von Greathouses und Charakteren verantwortlich ist. Im Unterordner Parser liegt eine Klasse, welche sich um das Parsen der Kommandozeilen Argumente kümmert. Im Unterordner Pathfinder befanden sich Hilfsklassen, welche sich für die Spiel-Runden-Phasen von Relevanz sind. In einem weiteren Unterordner namens validation befinden sich Klassen, welche sich um die Validierung der Konfiguration kümmern. Innerhalb des Server-Moduls befindet sich ein Ordner ClientManager, welcher die Verwaltung der verschiedenen Arten von Clients handhabt. Im Ordner Konfiguration befindet sich Ordner für Konfigurationsdateien und zugehörige Schemas. Zudem befinden sich hier Klassen zum Laden und konfigurieren von Parteien und Szenarios. Im Ordner Parser befinden sich ein Kommandozeilen-Parser, welcher spezifische Kommandozeilenoptionen für den Server interpretiert. Im Ordner roundHandler befinden sich alle Klassen, welche für die Spiel-Logik und den Ablauf der Rundenphasen verantwortlich sind. Zudem befindet sich im Server-Modul eine Main Klasse, welche alle Startevents triggert, wenn der Server gestartet wird. Zu guter Letzt liegt noch eine Klasse ServerMessageController vor, welche sich um die Handhabung von serverseitigen Nachrichten kümmert.

3 KI-Client

3.1 Allgemeines

Dieser Abschnitt behandlet die Handhabung des KI-Clients, wobei eine hohe Ähnlichkeit zum Server besteht.
Das bedeutet, dass die Installation, das Bauen oder auch das Ausführen analog zu dem Vorgehen beim Server sind (siehe Abschnitt 2).

3.2 Kommandozeilenoptionen des KI-Clients

Der KI-Client besitzt alle im Standard aufgelisteten Konfigurationsmöglichkeiten(auch optionale Kommandozeilenoptionen). Weitere Kommandozeilenoptionen sind nicht vorgesehen.

Auflistung der Kommandozeilenoptionen des KI-Client:

address: Gibt die IP-Adresse des Servers auf dem gespielt wird an.

help: Listet alle verfügbaren Parameter auf. Falls individuelle Parameter hinzugefügt wurden so ist dieser Parameter verpflichtend.

port: Fordert nachfolgend eine (gültige) Portnummer, wird keine angegeben oder sollte dieser Parameter nicht unterstützt werden, so wird die Portnummer 10191 ausgewählt.

name: Fordert nachfolgend den Name des KI-Clients im Spiel, wobei dieser mindestens drei Zeichen lang sein muss.