Boardgame Simulator Anleitung

8. April 2021

Inhaltsverzeichnis

1 Kurzanleitung

Dies ist eine kurze Schritt-für-Schritt Anleitung um einen Server zu starten und vorhandene Spiele zu laden. Möchte man nur zu einem vorhandenen Server verbinden und ein neues Spiel laden, können die Punkte 2)-4) übersprungen werden. Wie man neue Spiele erstellt, wird in Kapitel 2 erklärt.

1) **Java installieren:** Für das Programm wird Java benötigt. Java kann unter folgendem Link https://www.java.com/de/download/ heruntergeladen werden und installiert werden.

2) **DynDNS Adresse erstellen und mit Router verbinden:** Mithilfe einer festen Adresse Adresse kann Verbindung zum Spiel vereinfacht werden. Solche Adressen können zum Beispiel auf der Seite https://www.noip.com kostenlos erstellt werden und sehen wie folgt aus *meinspieleserver.ddns.net*. Diese Adresse muss im jeweiligen Router hinterlegt werden. Je nach Routermodell kann das etwas unterschiedlich funktionieren!
   a) **Fritz-Box:** Link zum Tutorial
   b) **Telekom:** Link zum Tutorial

3) **Port freischalten:** Auf dem Router muss nun noch der Port freigeschaltet werden über den der Server läuft, damit die Mitspieler von außerhalb des lokalen Netzes auf den Server zugreifen können.

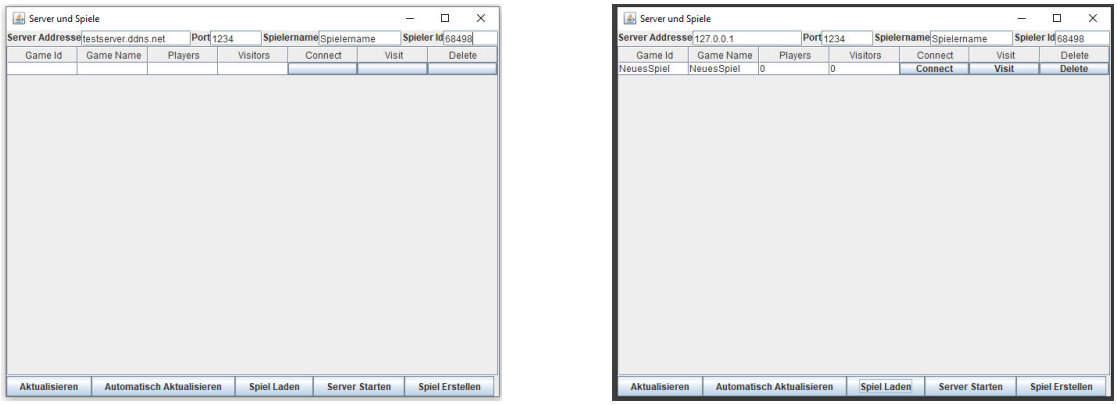
4) **Server starten:** Zuerst zum Programmpfad navigieren in dem die Datei SimpleBoardGameSimulator.jar liegt.
   a) **Windows:** Die Datei StartServer.bat, die im selben Verzeichnis liegt mit einem Texteditor öffnen und <port> durch den freigeschalteten Port ersetzen.
   b) **Linux:** Im Terminal den Befehl

      java -jar SimpleBoardGameSimulator.jar –server <port>

      ausführen. <port> muss durch den freigeschalteten Port ersetzt werden!

5) **Programm starten:** Das Programm SimpleBoardGameSimulator.jar ausführen. Nun sollte sich folgendes Fenster öffnen:

   Als Serveradresse muss nun die Adresse des Servers eingegeben werden (entweder die oben erstellte Adresse oder die Adresse eines vorhandenen Servers mit dem man verbinden möchte). Als Port muss der zuvor freigeschaltete Port eingegeben



(a) Startfenster des Spiels

(b) Startfenster mit geladenem Spiel

Abbildung 1.1: Startfenster des Spiels

werden oder der Port des vorhandenen Servers mit dem man verbinden möchte. Jeder Spieler kann sich einen Namen geben und sollte eine eindeutige Spieler Id wählen. Falls schon ein Spiel auf dem Server läuft kann “Aktualisieren” gedrückt werden um das Spiel anzuzeigen. Dann erscheint wie im Bild rechts das laufende Spiel und die Spieler können mit dem “Connect” Button verbinden.

6) **Spiel laden:** Wenn sich das Programmfenster geöffnet hat kann über den Button “Spiel Laden” ein neues Spiel, das man schon erstellt hat, geladen werden. Dazu navigiert man im Fenster, das sich öffnet zum Spiel (z.B. Spiel.zip) und klickt auf “Öffnen”. Das Spiel sollte nun für alle Spieler, die die richtige Adresse und den richtigen Port angegeben haben erscheinen, wenn sie auf “Aktualisieren” klicken. Klicken auf “Connect” lässt die Spieler mit dem Spiel verbinden.

2 Spiel erstellen

In diesem Abschnitt wird beschrieben, wie ein neues Spiel für den Brettspielsimulator erzeugt werden kann.

Jedes Spiel besteht aus Spielobjekten (Bildern im jpg oder png Format) und den XML-Dateien *game.xml* und *game_instance.xml*. Spielobjekte können drei verschiedene Typen haben card, figure und dice.

2.1 Die Datei game.xml

In der *game.xml* werden die Spielobjekte definiert. Das Grundgerüst der Datei sieht wie folgt aus.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xml>
<background>hintergrund.jpg</background>
</xml>
```

Das Bild mit dem Namen *hintergrund.jpg* wird zum Hintergrundbild des Spiels.
Zusätzlich können beliebige Spielobjekte durch den Tag

```
<object type="card"></object>
```

mit den unten genannten Typen definiert werden (in diesem Fall vom Typ card). Welche Attribute die verschiedenen Spielobjekte haben können wird in den entsprechenden Abschnitten erläutert.

2.2 Die Datei game_instance.xml

In der *game_instance.xml* wird eine Instanz des Spiels definiert, z.B. ob es einen Tisch geben soll oder nicht und an welcher Stelle die Objekte im Spiel zu Beginn liegen.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xml>
<settings>
        <name>Brettspiel</name>
        <table color="#d2a56d" put_down_area="true"
        table_radius="500">true</table>
</settings>
<object unique_name="object1" x="0" y="0" r="0"/>
</xml>
```

Das Beispiel definiert ein Spiel mit dem Namen *Brettspiel*, das ein Tisch mit Radius 500, der Farbe $\#d2a56d$, und einem Ablagebereich in der Mitte des Tisches hat. Das Spiel hat außerdem ein Objekt mit dem Namen *objekt1*, welches in *game.xml* definiert wurde und an Position $0, 0$ und mit Anfangsrotation 0 gezeichnet wird.

Unter dem Tag *settings* können alle wichtigen Spieleigenschaften definiert werden. Unter dem Tag *object* werden die Objekte mit Positionen, Rotationen usw. definiert.

Tagname	Attribute	Werte	Erklärung
name	–	String	Name des Spiels
table	color, put_down_area, table_radius	true/false	Anzeige des Tisches auf dem Spielfeld, falls False wird der Tisch nicht angezeigt, color definiert die Farbe des Tisches, put_down_area ist ein boolsches Attribut und gibt an, ob ein Ablagebereich in der Mitte des Tisches erscheinen soll, table_radius gibt den Radius des Tisches an.
private_area	–	true/false	gibt an, ob der Handkartenbereich zum Spielstart eingeblendet sein soll
seats	–	<seat color="#0000ff/>	Feste Liste von farbigen Stühlen um den Tisch
debug_mode	–	true/false	Falls true, werden zusätzliche Informationen angezeigt, die hilfreich beim debuggen sind

Tabelle 2.1: Mögliche Spiel Settings

2.3 Spielobjekttypen

Die Spielobjekte werden in der *game.xml* definiert. Einzelne Instanzen von Spielobjekten werden in der *game_instance.xml* definiert und mit einem eindeutigen Namen mit den Objekten aus der *game.xml* verknüpft. Spielobjekte werden durch den Tag *object* markiert. Alle Spielobjekte haben die folgenden Attribute.

Attribute	Erklärung
type	Typ des Spielobjekts
unique_name	Eindeutiger Name des Objekts
width	Breite des Objekts
height	Höhe des Objekts

Weitere Attribute für card, figure, dice Objekte können in den jeweiligen Abschnitten gefunden werden.

Instanzen von Spielobjekten können in der *game_instance.xml* mit folgenden Attributen definiert werden:

Attribute	Erklärung
unique_name	eindeutiger Name aus der *game.xml*
x	x Position zu Spielbeginn
y	y Position zu Spielbeginn
r	Rotation
s	Skalierung
number	Anzahl der Objekte des Typs
is_fixed	true/false, falls true kann Objekt im Spiel nicht bewegt werden

Beispieldefinition von Instanzen des Objekts mit dem Namen *objekt1* in der *game_instance.xml*:

```
<object unique_name="objekt1" x="0" y="0" r="0" s="1"
number="3" is_fixed="false"/>
```

In diesem Beispiel werden 3 Instanzen vom Objekt mit dem Namen *objekt1* an der Position $0, 0$ mit der Rotation 0 und ohne Skalierung erzeugt.

2.3.1 Der Typ card

Objekte vom Typ card sind Spielkarten. Sie können z.B. gestapelt und gemischt werden und haben eine Vorder-und Rückseite. Beispiel eines card Objekts mit Vorderseite *front.jpg*, Rückseite *back.jpg*, dem Wert 10 hat und in 45 Grad Schritten gedreht werden kann und zur Kartengruppe *Spielkarte* gehört:

```
<object type="card" unique_name="card1" value="10"
 rotation_step="45" front="front.jpg" back="back.jpg">
 <group>Spielkarte</group>
</object>
```

Attribute	Optional	Erklärung
front	Nein	Dateiname der Vorderseite (jpg, png)
back	Nein	Dateiname der Rückseite (jpg, png)
value	Ja	Wert der Karte
rotation_step	Ja	Mögliche Rotationen der Karte

Objekte vom Typ card können als Wert eine Gruppe erhalten über die sie eindeutig identifizierbar und sammelbar sind.

```
<group>Spielkarte</group>
```

2.3.2 Der Typ figure

Objekte vom Typ figure sind Spielfiguren. Sie können z.B. stehen oder liegen und auf Karten gestellt werden.

Beispiel eines figure Objekts:

```
<object type="figure" unique_name="objekt1"
 standing="standing.jpg" width="100"
 height="100"/>
```

Attribute	Optional	Erklärung
standing	Nein	Bild für die stehende Spielfigur (jpg, png)

2.3.3 Der Typ dice

Objekte vom Typ dice sind Würfel. Sie haben mehrere Seiten und können gewürfelt werden.

Beispiel eines dice Objekts mit sechs Seiten:

```
<object type="dice" unique_name="dice1" width="30" height="30">
        <side value="1">side1.jpg</side>
        <side value="2">side2.jpg</side>
        <side value="3">side3.jpg</side>
        <side value="4">side4.jpg</side>
        <side value="5">side5.jpg</side>
        <side value="6">side6.jpg</side>
</object>
```

Objekte vom Typ dice haben als Werte Seitenobjekte mit Tag *side*, mit einem Attribut value und als Wert das Bild der Seite.

2.3.4 Der Typ book

Objekte vom Typ book können wie Bücher verwendet werden. Sie haben mehrere Seiten und können vor-und zurück geblättert werden.

Beispiel eines book Objekts mit sechs Seiten:

```
<object type="book" unique_name="book1" width="30" height="30">
        <side value="1">side1.jpg</side>
        <side value="2">side2.jpg</side>
        <side value="3">side3.jpg</side>
        <side value="4">side4.jpg</side>
        <side value="5">side5.jpg</side>
        <side value="6">side6.jpg</side>
</object>
```

Objekte vom Typ book haben als Werte Seitenobjekte mit Tag *side*, mit einem Attribut value und als Wert das Bild der Seite.

3 Spielsteuerung