Universität Leipzig - Softwaretechnik Praktikum 2014/2015
Entwurfsbeschreibung zum Projekt: Ein kartenbasiertes "Multiplayer"-Spiel

Gruppe: SWT15-GKP

26. Mai 2015

Inhaltsverzeichnis

1 Allgemeines **2**

2 Produktübersicht **2**

3 Grundsätzliche Struktur- und Entwurfsprinzipien **3**

4 Struktur- und Entwurfsprinzipien der einzelnen Pakete **3**

4.1 Server . . . 3

4.2 Client . . . 3

4.2.1 Map Modul . . . 3

4.2.2 Game Modul . . . 4

4.2.3 Highscoremodul . . . 5

4.3 Bibliotheken . . . 5

5 Datenmodell **5**

6 Testkonzept **5**

7 Glossar **6**

1 Allgemeines

Im Rahmen des SWT-Praktikums wird ein kartenbasiertes Multiplayerspiel entwickelt, das es ermöglicht, ein an das alte Pacman angelehnte Computerspiel auf einem realen Kartenausschnitt zu spielen. Die entstehenden Highscores werden dann zu den jeweiligen Karten gespeichert um sich mit anderen Spielern messen zu können.

2 Produktübersicht

Der Nutzer kann online über die Spiele-Website auf das Programm zugreifen. Dabei stehen dem Nutzer ein Suchfeld zum finden der gewünschten Spielumgebung zur Verfügung, es kann auch manuell durch Scrollen über die Karte ein Spielort ausgewählt werden, wobei nur Städte ab einer bestimmten Größe zugelassen sind, damit auf jeden Fall eine spielbare Karte erstellt werden kann und die Highscore vergelichbar bleibt. Hat man den Ort ausgewählt, wird das Spielfeld erzeugt und man kann ein neues Spiel beginnen. Man kann den Pucman nun mit Hilfe der Pfeiltasten steuern. Ziel des Spiels ist es den Pucman über die Karte zu steuern und die Kekse die auf der Karte verteilt liegen zu fressen. An dieser Aufgabe wollen den Spieler Geister hindern, die sich auch über das Spielfeld bewegen und Pucman fressen können. Frisst ein Geist den Spieler so wird dieser auf den Startpunkt zurückgesetzt und die Anzahl seiner Leben wird um eins verringert. Es gibt unterschiedliche Geister, die durch ihr aussehen unterschieden werden können und sich anhand verschiedener Logiken bewegen. Falls keine Leben mehr übrig sind, wird das Spiel beendet und die erreichte Highscore zusammen mit der Karten URI und dem Namen des Spielers abgespeichert. Während des Spielens werden die Möglichkeiten zur Veränderung der Kartenansicht deaktiviert. Das Spielgeschehen ist durch einen Soundtrack unterlegt, der eine funktionale und inhaltliche Verbindung zwischen Bild und Ton generiert. Desweiteren werden für das Spiel wichtige Informationen wie aktueller Highscore, die verbleibende Anzahl an Leben und die Möglichkeit direkt auf die Website des Spiels zugreifen zu können angezeigt.

3 Grundsätzliche Struktur- und Entwurfsprinzipien

Als grundsätzliche Programmiersprache haben wir uns für Java-Script entschieden. Wobei der Java-Scriptcode von einer HTML-Seite aufgerufen wird. Desweiteren benutzen wir Phaser als Gameframework.

Ein Teil der Software wurde mithilfe der Java-Script Bibliothek JQuery geschrieben, da es keine adäquaten Funktionen in Java-Script gibt. Die von uns umgesetzte Web-Anwendung basiert auf einer Client-Server Architektur. Wir arbeiten also mit einem HTML-Server auf dem die Daten liegen, die vom Client abgefragt und an diesen übertragen werden. Diese Daten beinhalten auch den Gameblock, der dann vom Client ausgeführt wird. Außerdem gibt es noch ein Map-Module, welches das Level aus den Geo-Daten erstellt, die mithilfe der Linked Geo Data API von OpenStreetMaps bezogen werden. Dazu kommt das Highscore-Module, welches für die Verwaltung der Highscores verantwortlich ist.

Desweiteren kommt noch ein Datenserver hinzu, der die Highscores unter Verwendung eines Tripplestores speichert.

4 Struktur- und Entwurfsprinzipien der einzelnen Pakete

4.1 Server

Vom Server werden die benötigten Spieldateien zur Verfügung gestellt und automatisch geladen. Desweiteren werden Highscoreanfragen und Manipulationen zwischen dem Dataserver und dem Client verarbeitet und anschließend weitergeleitet.

4.2 Client

4.2.1 Map Modul

Die Linked GeoData API extrahiert aus OpenStreetMap die Geo-Daten für das Erstellen der Hintergrundkarte in Abhänigkeit vom gewählten Ort.
Die Overpass API extrahiert die Geo-Daten aus OpenStreetMap, die zum Hervorheben der benutzbaren Straßenzüge und zur Erstellung des Levels benutzt werden.
Die Datei **Graph.js** sorgt dafür, dass aus den Geo-Daten von OpenSteetMap weiterverarbeitet werden. Die Erstellung des Spielgraphen ist in dieser Software, der aufwendigste Teil.
Durch eine Vorselektierung durch eine angepasste Query für die Overpass

API werden die ersten Straßenzüge entfernt, wenn diese bestimmte Kriterien nicht erfüllen, so wurden z.B. Fußwege, Fahrradwege, aber auch Treppen entfernt, da das Straßennetz ansonsten nicht verwendbar gewesen wäre.
Auf dem bereitgestellten Graphen, werden nun verschieden Graphtheoretische Algorithmen angewendet um das Ergebnis spielbar zu machen.
Es wird der größte zusammenhängende Teilgraph gesucht, dieser in eine eigene Datenstruktur überführt, und eine Interpolation gestartet, die zwischen den Knoten des Graphens die fehlenden Pixel setzt, s.d. die Spielfigur, sich über ein Netzt aus Pixeln bewegen kann, die eine bestimmte Dichte ausweisen.

4.2.2 Game Modul

Der Gameblock wird an das Model-View-Controller Prinzip angelehnt, welches durch die Nutzung von Phaser umgesetzt wird. Phaser übernimmt für uns die Eingabe und Ausgabeverarbeitung und stellt den Gameloop zur Verfügung, dies ist eine Schleife, die währen des Spielablaufs immer wieder durchlaufen wird. Dabei werden im wesentlichen zwei Dinge ausgeführt:
Die Update-Methode, enthält: Tastatur- und Mauseingabe, Bewegung, Kollisionserkennung, ...
Die Render-Methode, enthält: Anzeige aller Sprites an den neuen Positionen, Hintergrund animieren, ...
Die Game Loop wird abgebrochen, wenn das Spiel beendet wird.

Interface.js
In diesem Modul wird die Schnittstelle zum Benutzer definiert. So werden hier z.B. Knöpfe und Anzeigen, die dem Benutzer Informationen über den aktuellen Spielstand liefern, bzw. diese Abrufbar machen erstellt. Die Schnittstelle ist der Teil eines Systems, welcher der Kommunikation dient.

Load.js
Zeigt einen neuen Ladescreen, während die Karte lädt.

Game.js
Stellt den Gameloop dar.

Character.js
In diesem Modul wird die Speilfigur, also der Pucman definiert.

4.2.3 Highscoremodul

Das Highscore Module verwaltet die Highscores der Spieler, d.h. es stellt Anfragen an den Server zum Triplestore und kann sie über den Server auch abfragen, falls sie im Spiel angezeigt werden sollen. Die Highscores werden in einem Tripelstore gespeichert, welche mithilfe von SPARQL Queries abgefragt werden, bzw. durch SPARQL Update Methoden in die Datenbank eingepflegt werden.

4.3 Bibliotheken

Cytoscape.js
Diese Bibliothek wird von dem Map Modul benötigt um die Graphalgorithmen auszuführen. In der Bibiothek sind bereits viele Algorithmen enthalten.

JQuery.js
Diese Bibliothek wurde vor allem benötigt um Http Anfragen zu realisieren.

OsmtoGeoJson.js
Diese Bibliothek wird auch von dem Map Modul benötigt, um Geodaten aus dem OSM Format in das GeoJson Format zu übertragen.

Phaser.js
Wie betreits erwähnt stellt Phaser als Gameframework einen sehr wichtigen Teil unseres Projektes dar, die Bibliothek stellt alle Funktionen zur Verfügung die man als Spieleentwickler benötigt um einen flüssigen Spielefluß, eine Eingabe- und Ausgabeverarbeitung etc. zu haben.

5 Datenmodell

Siehe Abbildung 1.

Beschreibung zu Abbildung 1:
$A \xrightarrow{\text{Datenfluß}} B$ beschreibt den Datenfluss von A nach B, wobei A und B Module der Software sind.

6 Testkonzept

Am Ende des Sprints wird die aktuelle Version der Software auf den Webserver geladen. Diese wird anhand eines Test-Workflows geprüft. Dabei werden die angegebenen Eingaben getätigt und die Reaktion der Software mit der erwarteten Ausgabe verglichen. Sollten dabei Abweichungen auftreten, gilt der Test als nicht bestanden.

7 Glossar

Das Glossar wurde überarbeitet und als externes Dokument angefügt.

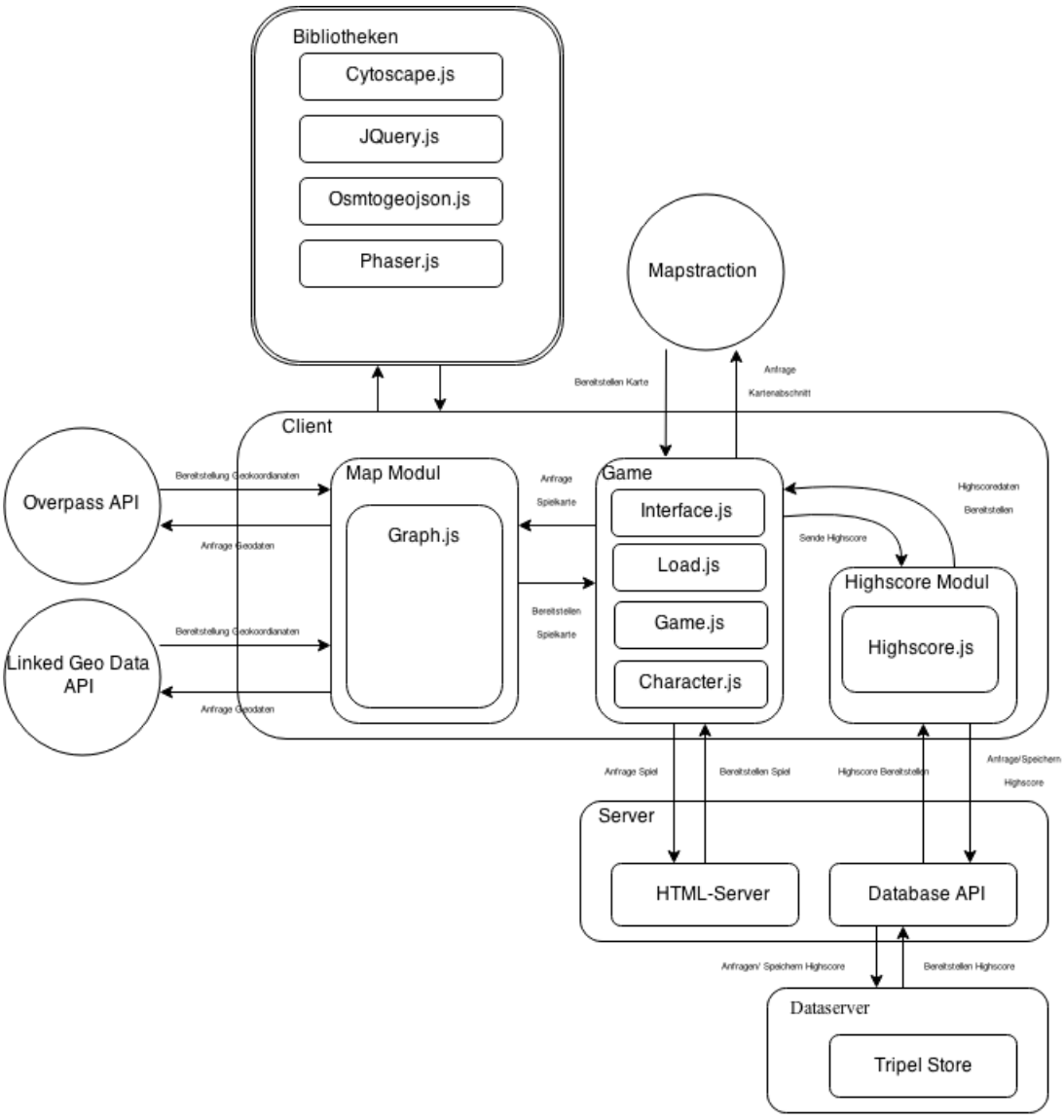

Abbildung 1: Architekur